Directed Computer Aided Design Using Expert Systems Techniques

A thesis submitted in fulfilment of the requirements for the Degree of
Master of Engineering in Mechanical Engineering
at the University of Canterbury,
Christchurch, New Zealand

by

Peggy Wenping Geng

July 1994

Acknowledgments

I would like to express my sincere thanks to my supervisors Dr Ken Whybrew and Dr Anne Ditcher, for their guidance and supervision, and their great degree of patience and understanding toward a foreign student with poor English, throughout the entire project process.

Very special thanks to Dean Shilvock, who gave me enormous assistance in editing my thesis and making it fully intelligible. I also wish to express my appreciation to my other fellow postgraduates, Simon Fry, Tim Jones and Bernd Nennemann, who were very supportive.

And I would like to thank Mr Phil Smith and Mr Julian Murphy, for their help and advice on the use of computers and software packages.

Finally, I would like to mention my sponsor, Clare Fearnley, my friends, Bridget Underhill, Margaret and Paul Rees, and Paul B. Jones who gave me invaluable support during the duration of my time in this beautiful country.

Abstract

The work reported here is an examination into the interfacing of software. The general objective of the study was to investigate the methods of communication between software packages in a local environment. The software packages discussed have been grouped into four areas: language; database/spreadsheet/word processor; CAD and CAM. The study specifically follows a design process to illustrate methods of collecting and transferring data between user and CAD/CAM software packages which deal with specific aspects in the design process, such as drawing, analysis and manufacturing. The design of a shaft and its associated components was deliberately chosen for investigation because shaft design is one of the most common steps in machine design. The shaft design assistant system was developed based on the MicroStation environment and most parts are written in MicroStation Development Language which is another area of interest explored in this study. Incorporated into the shaft design is the framework of a knowledge-based module which assists the designer in the selection of suitable bearings. This design process was developed in Prolog to provide an example of communication between MDL applications and programs written in other languages. This module also shows the basic concept of how knowledge based systems are applied to engineering component selection. The interface issues were addressed along with the development of the shaft design assistant system.

Contents

Chapter 1 Introduction

1.1 Objectives and Scope of the Study

Nowadays computers are widely used in every field including engineering. Computer-aided design and computer-aided manufacturing have expanded greatly along with the development of software and hardware. There is a number of ways in which computers can aid designers. These are: "performing design calculations, producing and managing parts lists, storing and retrieving design information, producing drawings, and producing programs for numerically controlled machine tools" (Peter F. Jones, 1992, p13). Computers are now routinely involved in every phase of design and manufacturing. Each phase of the design process can be related to the tools that CAD/CAM systems provide for engineers. Each tool is supported by programs and software packages. The software and programs introduced by different companies include drafting systems, analysis packages, word processing for document preparation, database management systems, manufacturing process planning systems and numerical control programs. Each software program performs specified functions in the design process to aid the designer and also has its own individual features to meet the requirements of its customers. Development of software packages has aimed at expanding their scope in the design and manufacturing process, however they still can not encompass the whole activity. There is a need for different software packages to communicate with each other so as to enable the use of facilities which the specialised CAD/CAM systems provide. Interfaces between the software packages are used to fulfil this function. The software

packages interact with each other via the interfaces and work cooperatively to complete an engineering task. The user interface channels the communication between the user and the particular CAD/CAM system.

The work reported here is an examination of interfacing software. The general objective of the study was to investigate the methods of communication between software packages in a local environment. There are several software packages installed in personal computers or workstations in the Department of Mechanical Engineering at the University of Canterbury. These disjoint systems can fall into four main categories: CAD systems which include drawing and analysis packages, such as AutoCAD, MicroStation, EMS, Algor, Dynapak, PipePlus, FEM, Nastran, Sysnoise, Fluent, etc; CAM systems which include numerical control programs and numerically controlled machine tools such as MasterCAM, Fanuc system, etc; database/spreadsheet/wordprocessor which deal with data and text documents such as Oracle, Excel, dBase, Microsoft Word, WordPerfect, etc; and computer languages such as C, Basic, Fortran, Prolog, etc. The main packages are listed in Appendix G. These packages supplied by different companies deal with various activities during the design and manufacturing process and are isolated or loosely integrated. This investigation primarily concerned interfacing and the specifications of packages currently in use for transferring information. Most data formats which are used in the packages are incompatible. Several packages embed some data transfer facilities. The communication between these packages is via two main methods: direct transfer and specified data format transfer. Direct transfer is when the data used in one package can be transferred to another without any change while specified data format transfer is when the interface can be achieved in a neutral

format, such as DXF, IGES or other data format, like PLT etc. The main connections between packages are described as follows.

CAD group: This group embraces several graphic software packages such as MicroStation Version 4 and AutoCAD Version 10 for PC, EMS for workstation and some analysis packages, like Algor, PipePlus, Dynapak, Nastran, Sysnoise, Fluent, etc. The DGN format can access EMS directly because EMS has a DGN translator. The data transfer between MicroStation Version 4 and AutoCAD Version 10 can be obtained via DXF format. A graphics file can be exchanged between MicroStation Version 4 or Version 5 and Algor in DXF or IGES format. The 3D DXF translator in Algor only supports lines and arcs. Data transfer is carried out within PipePlus, Dynapak and Algor either in ESD with particular elements or DXF format. The packages within this group can also interface with packages in other groups eg. a graphics file can be sent to MasterCAM via DXF or IGES formats. The DXF translator receiving data supports elements like points, lines, arcs, circles, levels, blocks, etc. The plotting files can be transferred between MicroStation and wordprocessors such as WordPerfect and Microsoft Word in PLT format. Similar files from Algor into wordprocessors use HPGL. The screen capture functions in MicroStation create screen image files in several formats, like GIF, RGB, BMP, PCX, PICT, WPG, etc. These functions also can be used in Algor to create Bitmap files.

CAM group: MasterCAM can import and export some data formats like ASCII, CADL, DXF, IGES, NFL, and GEO. Numerical control files from MasterCAM can be transferred to the milling machine directly.

Database/spreadsheet/wordprocessor: Text files can be transferred between these packages such as Microsoft Excel, Microsoft Word, WordPerfect 5.1, DrawPerfect, dBase 4, Quatro Pro 3.0, VP Plan utilising various data format translators embedded in the packages.

Languages group: Some of the packages above have facilities to call functions or programs made in other languages for example MicroStation Development Language can call standard C functions and interface programs in other languages through its own communication functions. MatLab has channels to call C or Fortran subroutines.

A map related to the above investigation is drawn in Appendix G, it shows more interface details. Concerning the interfaces within popular CAD/CAM packages, the ideal is to develop an unified, integrated design environment that includes all of the specialised CAD/CAM software and a good-quality interface allowing the engineer to work with these packages in a consistent and concurrent way. However, the idea of developing this kind of environment based on the Windows system was abandoned due to the limitation in computer memory size which packages must use whenever they are active. This limitation lead to examination of communication between the designer and programs based on MicroStation because the MicroStation programming environment provides a means of interface which can access other programs and communicate with the designer. The study specifically follows a design process which includes the basic activities occurring in an engineering design task. It is aimed at illustrating methods of collecting and transferring data between user and software packages related to design and manufacturing activities. The design of a shaft and its associated components

was deliberately chosen because shaft design is one of the most common steps in machine design. The shaft design assistant system was then developed based on the MicroStation environment. Some of topics were addressed along with development of shaft design assistant system such as user interface, data management, geometry modelling, design analysis and manufacturing and artificial intelligence. The system is written in MicroStation Development Language and Prolog computer language.

The thesis presented here is organised as follows: firstly, the general information and background related to the study are provided. This is followed by a literature review of development of CAD/CAM systems. From Chapter 3 to 7, the techniques and methods which were used to program modules and perform functions in the fields of interface design, database management, feature based design, program communication and knowledge-based systems are discussed. The interface issues are described in detail throughout these chapters. The shaft design example is then used to give a sense of the system implementation. Finally, limitations of the systems are described, conclusions are drawn, and further development points are discussed.

1.2 Program Environment

Geometric modelling is a basic part of computer-aided design and manufacturing. It describes the objects in an analytical, mathematical and abstract manner. MicroStation is a computer graphic software package. It has facilities as a computer aided design system to create 2D and 3D models. It

also enhances the extensive use of the MicroStation CAD engine by providing three application development facilities. They are: user commands (UCMs), MicroStation Customer Library (MicroCSL) and the MicroStation Development Language (MDL).

MDL is a complete development environment that lets applications take full advantage of the power of the MicroStation CAD engine. Actually, MDL is the application engine for MicroStation and provides all of the tools needed to create applications, "ranging from simple utilities or customised commands to sophisticated commercial applications, with a look and feel that is consistent with MicroStation's graphical user interface" (Intergraph,1991,M1-1).

"MDL is a very complete implementation of the C language, complying with the ANSI standard" (Bill Steinbock,1991, p5). The construction of MDL is listed in Appendix A. However, the following characteristics of MDL are worthy of address here.

In addition to a full range of functions available with any C compiler, MDL provides access to a large runtime library with over 1200 functions. Most of these functions are built-in functions, which means that they are functions embodied in MicroStation and MDL applications may call these functions directly (Intergraph,1992,p1-1). The function categories cover the whole range of graphic elements, creation and manipulation, view and state control, input and output and so on. Therefore, MDL provides the application with geometric-modelling and other aids which facilitate the development of geometric-modelling, design and analysis software.

A good-quality user interface is a common requirement of the interactive CAD/CAM system. MDL provides designers with tools to implement a graphical user interface for the application programs. This is an unique characteristic of MDL. Incorporating input and output functions, the graphic user interface represents the means of communication between the user and the program.

Finally MDL has facilities to call and communicate with external programs. This enhances the capabilities of MDL applications.

For the reasons discussed above, MDL has been selected as the main program development environment for this study.

There are several methods to develop knowledge-based CAD/CAM software. The details will be discussed in Chapter 7. In this study, the Prolog language is used to implement the techniques of artificial intelligence. The language of Prolog, which is stands for programming in logic, is based on mathematical modules of human thinking or deliberate reasoning (Steven H. Kim,1991,p2).

1.3 Shaft Design Process

A shaft refers to a rotating member, of round cross section, used to rotate and transmit power. Elements such as gears, pulleys, flywheels, cranks, sprockets, and the like are usually attached to shaft which provides the axis of rotation, or oscillation of these elements. A shaft design begins after some

preliminary work, in which the elements attached on the shaft have at least been partially analysed and their size and spacing tentatively determined (Shigley, 1983, p677). Only the simple principles of shaft design were considered in this investigation as the shaft serves only to illustrate the main features of this investigation into interfacing methods. Fig.1.1 shows the steps in the shaft design process.

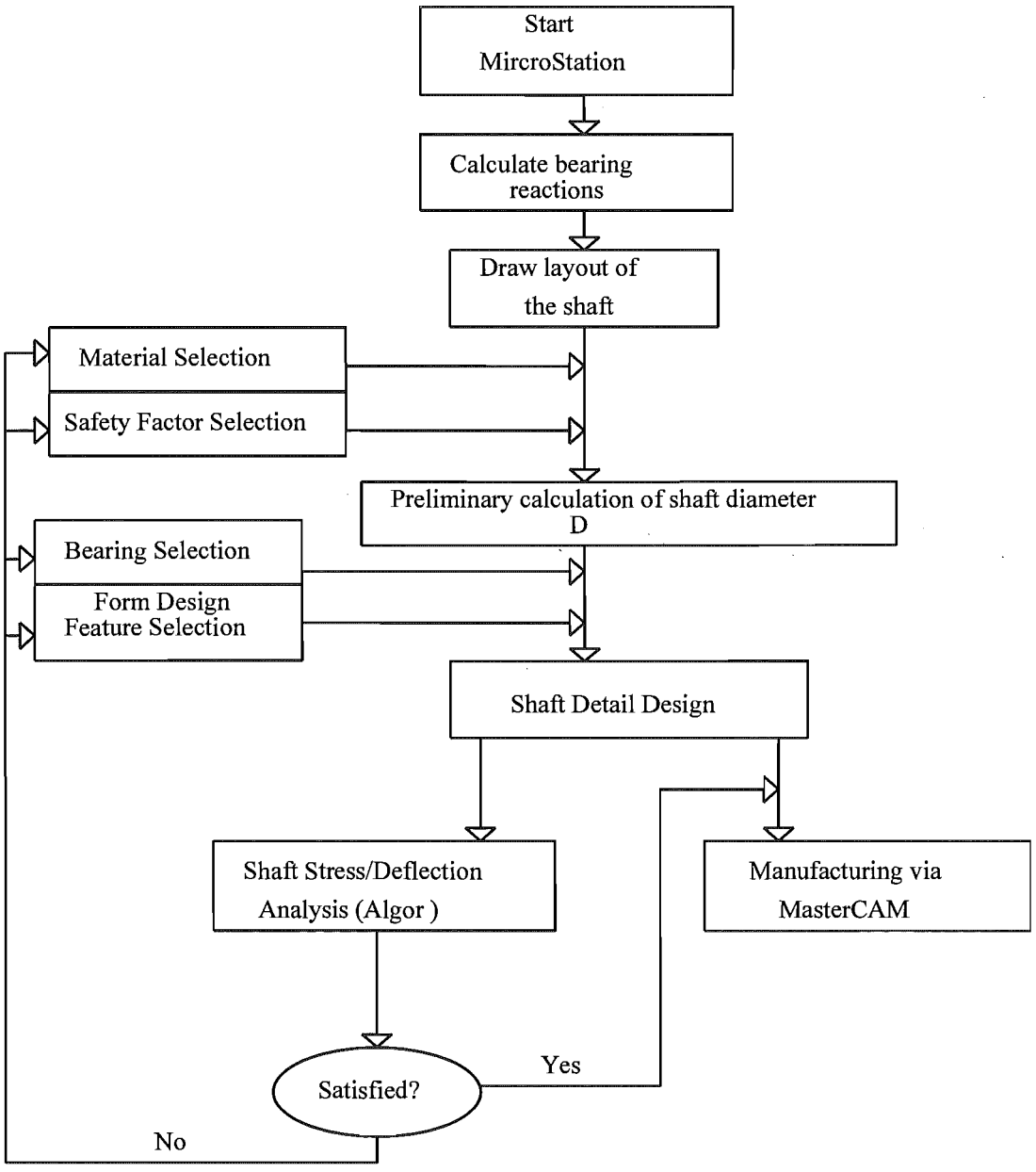


Fig.1.1 Steps in Shaft Design

The shaft design in this study starts with the basic known specification, such as input elements, output elements, the force and moment caused by the element and shaft configuration. Design actions then go through the following stages.

1. Calculate Bearing reaction forces;
2. Find the worst case position along the shaft, select a material and a safety factor and calculate a preliminary shaft diameter based on failure under peak static load (Appendix G1);
3. Make a preliminary bearing selection;
4. Design the shaft by detail - shaft detail design by features;
5. Shaft stress / deflection analysis;
6. Repeat the above process by changing materials, bearings, shaft dimensions, form design, etc;
7. Transfer the shaft geometry file to MasterCAM for a manufacturing solution.

1.4 Program Architecture

As mentioned earlier, the program is written in MDL and Prolog and mainly runs within the MicroStation environment. The overall structure of the program is shown in Fig.1.2. The program contains ten main modules. Each module performs a specified design action in the shaft design process. The user can interface with each module through the graphical user interface. The modules are listed by function as follows:

Input data module: accepts and manipulates the shaft force and torsion input data.

Bearing reaction calculation module: contains routines for the calculation of bearing reaction forces.

Draw a shaft sketch module: contains the routines to draw the shaft layout with dimensions included.

Material selection module: this module selects materials from the material database.

Shaft diameter calculation and drawing module: contains the routines for calculating the shaft diameters based on failure under peak static load and then the drawing of the shaft geometry.

Shaft detail design modules: contains the routines to manipulate and manage shaft feature data.

Bearing selection module: contains the routines to assist designer to select a bearing type and size for supporting the shaft.

External program communication module: provides the channel to interact with the bearing selection module.

Key selection module: this provides a channel for selection of a key from the key database allowing design of the keyway geometry.

Shaft draw module: contains the routines for drawing a shaft in terms of assembling the shaft features.

DXF or IGES file transfer module: provides the converter that transforms the shaft geometry file to a DXF or IGES format file.

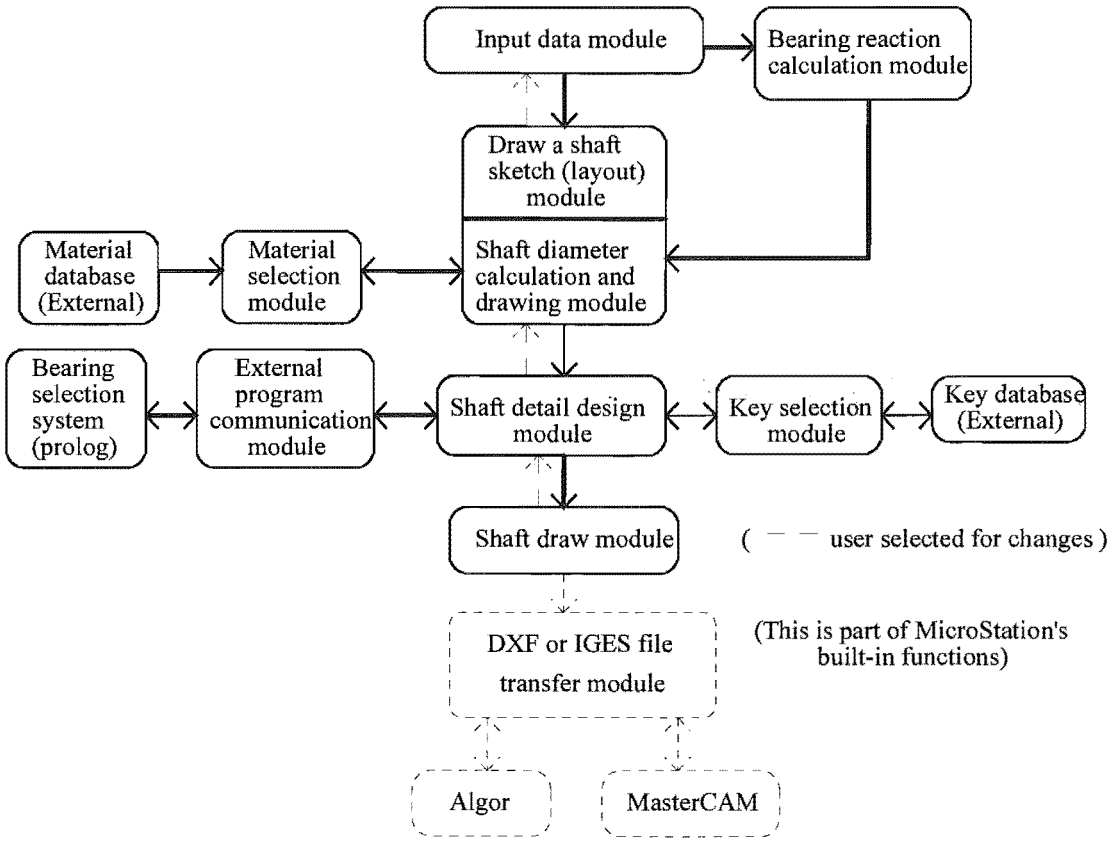


Fig. 1.2 Structure of the Program

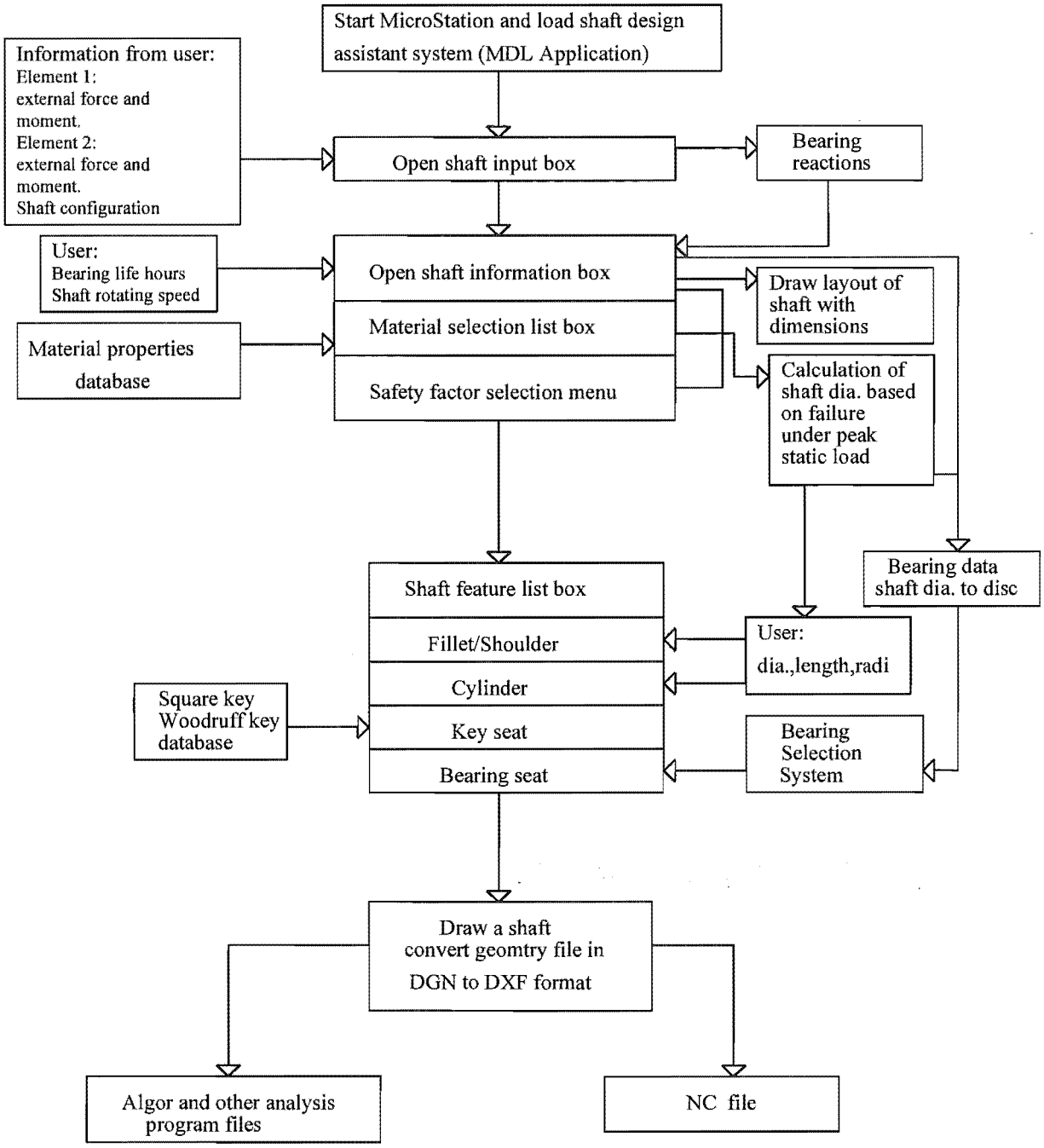


Fig.1.3 Information Flow Chart for Program

Fig.1.3 shows the information flow chart for the modules. The data communication between the modules is made either through the global variables within the program or through the files on the disc.

Chapter 2 Literature Review

This study focuses primarily on interfacing software incorporating shaft design processes as in a local environment. The concepts and techniques which are used in this approach have concentrated on the diverse aspects of CAD/CAM, such as user interface, CAD data transfer, feature technology and knowledge based systems. This chapter surveys the recent research related to these fields.

2.1 User Interface

The design of user interface serves as a basis to the success of a CAE system. Programming user interfaces is difficult and expensive, and research has concentrated on developing software tools such as user interface management systems (UIMS), user interface toolkits and interface builders to help with construction and maintenance of the user interfaces.

User interface toolkits and interface builders basically contain libraries of user interface components which can be composed into user interfaces. One of the examples is the user interface builder named "MOG". The user interfaces based on this approach have "in-built editing capability and are inherently tailorable" (A. Coleboune, P. Sawyer and I. Sommerville, 1993, p1). Some of the toolkits are designed specially for non-programmers, such as

"QUICK", to build high quality interfaces. The major characteristic of the "QUICK" system is its maximising power and flexibility within a very simple environment (S. Douglas, E. Doerry and D. Novick, 1990).

Along with the popular use of graphical user interfaces in recent years, the builders and toolkits which aid the construction of graphical user interfaces were developed. Pausch, Conway and DeLine (1992) produced "SUIT" which provides the fundamental components of an interface builder and graphical user interface toolkit and explanations with these components. The aim of the system is to let designers create a graphic user interface without spending much time becoming proficient (R. Pausch, M. Conway and R. DeLine, 1992). Another graphical user interface builder named "Lapidary", which refers to Lisp-based Assistant for Phototyping Interactive Designs Allowing Remarkable Yield, provides tools for designers to interactively specify all graphical aspects of an application and also provides a drawing editor allowing the designer to manipulate objects used in interfaces with a pre-defined library (B. Vander Zanden and B.A. Myers, 1991).

User interface management systems (UIMS) are one of the new basic approaches that help with the task of building user interfaces. A UIMS is defined as "a software system that implements some or all the interfaces between the user and the application's action routines that are involved when the user enters commands" (Francis Neelamkavil, 1989, p60). A number of user-interface management systems are available and each one has its own characteristic and speciality. Sastry discussed the main features of three of these systems according to the perspective of engineering applications user interface development and came to the conclusion that "Graphic Modelling

System", "TeleUSE" and "USEIT" are recommended specifically to engineering applications (L. Sastry, 1992). Singh, Kok and Ngan developed a UIMS named "Druid" to facilitate the design of interactive, graphical, direct-manipulation user interfaces (G. Singh, C.H. kok and T.Y. Ngan, 1990). "VUIMS" is an object-oriented user interface management system, which is one of the approaches based on the design methodology used, to "support reconfigurable components" used in user interfaces (J.H. Pittamn and C.J. Kitrick, 1990, p1).

The techniques of artificial intelligence are also used to facilitate the user interface design process. St. Jacques, Stevens, Getchius and Lau introduced the knowledge engineering principles and knowledge-based software tools to the user interface design. The approach based on the above tools and "method of representing complex navigational logic" was proven to be more efficient (M. St. Jacques, D. Stevens, J. Getchius and L. Lau, 1992, p1).

2.2 CAD Data Exchange

With the development of CAD/CAM systems and techniques the CAD/CAM data exchange among systems is an increasingly important problem. This leads to the data exchange standards which have been developed to help communications between the CAD/CAM systems.

The IGES (Initial Graphics Exchange Specification) format is the most widely used neutral format for the exchange of CAD/CAM/CAE data between

dissimilar systems and also serves as a link between the CAD and CAPP systems. Seemadula developed a process planning system using the IGES interface to transfer the geometric data from the CAD system (M.E. Seemadula,1993). Kalta and Davies developed "CADEXCAP" as a media to link CAD system and "EXCAP" which is a knowledge-based process-planning system via IGES (M. Kalta and B.J. Davies, 1993). Madurai and Li developed an expert system for automatic extraction and recognition of part features to connect CAD and CAPP. The system used IGES as its input data format (S.S. Madurai and Lin Li, 1992). Kumar, Anand, Anjanappa and Kirk presented an intelligent feature extraction methodology which reads CAD data transferred from any CAD system in IGES format (B. Kumar, D.K. Anand, M. Anjanappa and J.A. Kirk, 1993).

There have been some other techniques developed to help reading and transferring of IGES files. Bradbeer developed an IGES editor using Ada and Ada Design Methodology (P.E. Bradbeer,1992). Kalta and Davies have worked on a software package named "readiges" to restore IGES files in more conveniently accessible direct-access files (M. Kalta and B.J. Davies, 1993).

One of the standards for product data exchanges is "STEP" (Standard for Transfer and Exchange of Product Model Data) defined by the International Standards Organisation (ISO). "STEP" is targeted to represent product data in "a consistent and unambiguous manner" (Kiggans, 1991, p1) and is supposed to be "the neutral format by which product definition data will be represented worldwide" (Hemnelgarn and Hodges, 1991, p1). The project named "CADEX" was a development to provide the valid "STEP" processors for several CAD and FEM systems in the early versions of "STEP".

Helpensten and Heinrichs presented the structure of "CADEX". "CADEX" processors have a common architecture with an intermediate data structure which reflects the original "STEP" data structure. Solid models in boundary representation and constructive solid geometry, surface models with sculptured and analytical geometry, wireframe models and compound boundary representation models for the exchange of FEM appropriate topology are included in the system (H.J. Helpensten and H. Heinrichs, 1992). Kojima, Nakamura, Kugai and Kimura have developed the "EXPRESS" system to describe the specifications in data exchange processes in terms of "a "STEP" application protocol, CAD data schema definition and data conversion functions" (T. Kojima, I. Nakamura, Y. Kugai and F. Kimura, 1993, p1). "STEP" data input and output programs are generated as generic while CAD data input and output programs which are CAD system dependent are generated separately. The approach described in the paper is implemented in adherence to the "STEP" standard (T. Kojima, I. Nakamura, Y. Kugai and F. Kimura, 1993). Qiao, Zhang, Liu and Wang developed a product data preparation procedure for CAPP which uses "PDES/STEP" (Product Data Exchange Specification) as its foundation. (L.H. Qiao, C. Zhang, T.-H. Liu and H-P.B. Wang 1993). KirK, Liu and Fischer presents "an information integration model" which is based on "a global-local model scheme and the PDES/STEP" for the integration of CAE and CAM applications (J.W. Kirk, Thu Hua Liu and G.W. Fischer, 1992, p1). Liu and Fischer also developed an object-oriented approach to implement the PDES/STEP-based information model mentioned above for the feature-based manufacturing applications (Thu-Hua Liu and G.W. Fischer, 1993).

Some systems have concerned several data exchange formats. Helpenstein discussed methods for integrating the CAD and FEM systems. The approach is through a neutral interface. The DXF's (external files), VDAF's (national industrial sector identity) and STEP (international product data exchange standard) interfaces were recommended (H.J. Helpenstein, 1993). Klueh and Cashman examined several industry standards like IGES, SET (Standard d'Echange et de Transfert, Normlisation Francaise Z68300), VDA(Used in the German automobile industry) and PDES /STEP (Product Data Exchange Specification) as interfaces between mechanical design automation applications (D.W. Klueh and J.E. Cashmean, 1991).

Along with the introduction of the STEP, Hemmelgarn and Hodges discussed the one function of PDES Toolkit Software called IGES to STEP translator (ITOS), which can migrate the IGES to STEP (D. Hemmelgarn and J. Hodges, 1991).

Other than data exchange standards, there is other research focusing on the methods for CAD data transfer. Kiruchi, Kishinami and Saito have presented an approach for automatic CAD data exchange. The approach is founded on a data format consisting of three parts. They are: "data structure, algorithms and data". "An application schema, conceptual schema and physical file data schema" form three schema architecture which is used in the approach (Y. Kiruchi, T. Kishinami, K. Satto, 1992, p1). An experimental system based on the approach was developed (Y. Kiruchi, T. Kishinami, K. Satto, 1992). Lalande proposed an idea to effectively transmit engineering data in a global automotive environment. The effective CAD data exchange is related to the issues of standards and unified operating procedures. Each

original equipment manufacturer has an internally developed system and/or a combination of "off the shelf" system. The author explained the thought via outlining the strategic or primary CAD/CAM system (M. Lalande, 1991).

2.3 Feature-based Design

There has been much research related to feature techniques and their applications in design and manufacturing. Salomons, van Houten and Kals presented a paper which overviewed the research in feature-based design up to 1991 (O.W. Salomons, F.J.A.M. van Houten and H.J.J. Kals, 1993). This survey on feature-based design will concern some research after 1991.

According to Salomons, van Houten and Kals (1993), there are three approaches toward obtaining features: feature recognition, design by features and interactive feature definition. It is currently believed that future CAD/CAPP systems should provide for both feature recognition and design by features rather than the above approaches alone. Laakko and Mantyla described a feature-modelling system which provides a hybrid of feature-based design and feature recognition in a single framework (T. Laakko and M. Mantyla, 1993). Lenau and Mu discussed the problems of CAD/CAPP integration and interface in the approaches, that are feature recognition and feature based design, to link CAD and CAPP. They introduced a design oriented manufacturing process database (NADED) and feature-based CAD and CAPP system (T. Lenau and L. Mu, 1993).

There are several research issues related to feature-based design mentioned in Salomons's paper. One is called multiple views. Normally, there are different views of the same form-feature for one component among design, analysis, manufacturing and so on. These different views are referred to multiple views on features. Salomons, Kappert, van Slooten, van Honten and Kals proposed a new approach in feature based design focusing on the link between CAD and CAPP systems. The solution to the multiple views proposed is that the manufacturing feature is formed by considering the different form features at component level and using abstract features and assembly relations (O.W. Salomons, J. H. Kappert, F. van Slooten, F. J. A. M. van Houten, H. J. J. Kals, 1993). Some research has been done on features in engineering analysis. Unruh and Anderson described an approach to reasoning from a feature-based model to finite element analysis model (V. Unruh and D.C. Anderson, 1992). Cavendish, Frey and Marin presented an approach to feature based design and feature-based mesh generation which proceeds from the level of a complete feature (J.C. Cavendish, W.H. Frey and S.P. Marin, 1991).

There is some research on the approach to feature based design (feature based modelling). Denzed and Vosniakos developed an object-oriented approach toward feature-based design. The system supports design in conceptual and detailed phases. The feature definition is described via parameters and procedures. The Boundary Representation (B-rep) and Constructive Solid Geometry (CSG) are used for feature geometry representation. The system provides a mechanism allowing specification of inter-feature relations (H. Denzel and G.-C Vosniakos, 1993). Hoffman and Juan proposed a "high-level, generative, textual representation named "Erep"

for feature-based solid modelling" (C.M. Hoffman and R. Juan, 1992, p1). Such representation is in a form that can be edited and transformed to any solid modelling system. Furthermore, the representation can be extended to a representation from which to derive analysis representations and process plans (C.M. Hoffman and R. Juan, 1992). Instead of using CSG methods and B-Rep schemes to represent design and manufacturing functions, Allada and Anand presented "octree and quadtree based representation schemes" in some manufacturing areas like robotic task planning, tolerance representation and inspection, feature based design and assembly modelling and verification (V. Allada and S. Anand, 1992, p1). Rao Nalluri and Gurumoorthy discussed the knowledge based approach for feature based modelling (S.R.P. Rao Nalluri and B. Gurumoorthy, 1992). Duan, Zjou and Lai presented Feature Solid Modelling Tool (FSMT) for feature based design and manufacturing (W. Duan, J. Zjou and K. Lai, 1993). They also developed Feature Oriented modelling Tool (FOMT) which consists of feature bases, knowledge bases and database (W. Duan, J. Zjou and K. Lai, 1991).

Feature-definition languages is one of the research issues related to feature based design. Rosen and Dixon discussed feature-based design and manufacturing languages based on "a language of form consisting of concepts, relationships and attributes" after examining the languages for the feature-based design of thin-walled components and languages for their manufacturability evaluation (D.W. Rosen and J. R. Dixon, 1992, p1).

Some research has concerned feature techniques in concurrent engineering. The topic covers feature generation (Hyowen Suh, S. Ahluwalia and J.E. Miller 1991), feature modelling (D. Xue and Z. Dong, 1993 ; E.

Molloy, H. Yang and J. Browne, 1993), computer-aided concurrent engineering systems (I. Horvath, P. Kulcsar and L. Horvath, 1993), feature-based representation conversions (D.W. Rosen and T.J. Peters, 1992), and feature mapping (J.-Y Kim, R.O. Mittal, R.M.P. O'Graly and E.E.Young, 1992).

2.4 Knowledge-based System

Over the past decade, artificial intelligence techniques and software systems have been developed to suit many areas of engineering design and manufacturing. The research survey here is limited to areas related to mechanical engineering design and manufacturing and concerns the recent two years only.

Kar Tshon Leong, Siang Kok Sim and Yin Wing Chan developed an integrated knowledge-based system for mechanical design. The design process for a mechanical design problem is divided into a set of tasks which can be subdivided into more smaller tasks in a hierarchy manner. These tasks are handled by "a hierarchy of integrated teams of specialists using a distributed problem-solving approach" (Kar Tshon Leong, Siang Kok Sim and Yin Wing Chan, 1991, p1). Dilger describes an expert system applied to mechanical engineering redesign. The design protocol is evaluated and the result of the evaluation is compared with the existing solution (W. Dilger, 1992).

Mihara, Hirose and Harada examined the unsolved problem of integration of CAD and CAM and proposed a solution in which a feature-based product modeller is linked to a manufacturing knowledge base (K. Mihara, A. Hirose and Y. Harada, 1992). Held, Jager, Kratz and Schneider integrated the knowledge-based systems with conventional CAD to enhance the ability of the CAD to handle the complex mechanical design tasks (H. -J. Held, K. -H. Jager, N. Kratz and M. Schneider, 1991).

Several researchers have concentrated on developing systems to assist design in different design phases. Duhovnik and Zavbi proposed a engineering design model for innovative design and implementation of an expert system for mechanical engineering design in the conceptual phase (J. Duhovnik and R. Zavbi, 1992). Chakrabarti, Bligh and Holden studied the main activities and problems which occur in mechanical embodiment design and then proposed a solution framework. A "hierarchical object-oriented knowledge representation scheme" is used to support its implementation (A. Chakrabarti, T.P. Bligh and T.Hokden, 1992, p1). Bertini described a expert system named "EMBMEC" which uses artificial inelegance techniques for the embodiment design of mechanisms and articulated systems (L. Bertini, 1993).

Lai described an expert system applied to product assembly. Upon analysis of the products the system makes suggestions for design refinement. The bell roller assembly is used as an example of the system (S.H.-Y. Lai, 1993). Oh, Sommerville, Taleb-Bendiab and French developed an expert system to improve mechanical assembly design (V. Oh, I. Sommerville, A. Taleb-Bendiab and M. French 1992). Chen and Wichman presented a

systematic approach to facilitate the design and planning of mechanical assemblies (C.L.P. Chen and C.A. Wichman 1992).

Abdou presented an integrated approach to generate a process plan. The process plan is drawn up through the knowledge based system in cooperating with the CAM database (G.H. Abdou, 1992). Dong and Hu discussed another approach which uses a manufacturing knowledge-based system to "generate feasible machining sequences for optimal process planning" (Z. Dong and W. Hu, 1991, p1).

Programs have also been developed to assist in the design and manufacturing of specific products. Cha, Rao and Zhao developed an integrated intelligent environment which assists gear manufacture (J. Cha, M. Rao and Z. Zhao, 1993). Haasis discussed a framework based on the knowledge-based design for implementing a composite design for the projection of gear transmissions (S. Haasis, 1993). Dzymek and Jung studied the development an intelligent computer-aided design system incorporating the design of electric motors (Z.M. Bzymek and Joo Jung, 1990). Negele and Rathke developed an interactive design expert named "KONEX+" for computer numerical control machine design. Motz and Haghighi developed a comprehensive mechanical design expert system which includes computer graphics, finite element analysis and design optimisation to implement a proposed knowledge based design module for mechanical components (D. S. Motz and K. Haghighi, 1991).

Programs were also involved in classifying and analysing the mechanical behaviour of materials. Capelo, Ironi and Tentoni developed a

system to reason automatically about visco-elastic materials (A.C. Capelo, L. Ironi and S. Tentoni, 1992, 1993).

Chapter 3 Interface Design

Charles Flurscheim defines interfaces as "the lines or surfaces that separate different states. In organisational terms, they are the lines of demarcation between different functions or departments. "In practice, they are the internal and external barriers across which information and ideas must pass" (Charles Flurscheim,1977,p1). The design of the user interface is fundamental in the CAD/CAM systems. This chapter introduces the general ideas of user interfaces and then details the techniques used in the MicroStation environment.

3.1 The User Interface

A user interface may be "visualised as a surface through which data and controls are passed back and forth between the computer and the user" (Francis Neelamkavil, 1989, p60). Generally speaking, a user interface acts as an medium that connects the user and the program and shapes the computer into a tool for a particular application. It encompasses those aspects of a computer system which a user experiences directly. User interface, in terms of physical aspects, includes the display devices, audio devices, printers, and input devices (Francis Neelamkavil,1989,p60). Research concerning the user interface has emphasised exploiting physical devices and the interaction techniques provided by them.

As a result of using mice, bit map displays and windows in the user interface, the graphical user interface has become popular in the application programs. In windows, menus and icons have been used with success to define the graphical user interface in terms of the visual appearance and function description.

A good interface is essential for good system performance, but programming a good user interface is a difficult task. There has been a growing effort to create construction tools to help with building user interfaces. The new development of approaches to the construction of user interfaces is introduced in Chapter 2. In this approach the main point of interest is based on methods to build up user interfaces in the local environment, MicroStation, which provides the powerful and rich tools needed for users to create graphical interfaces which are attractive and unique.

3.2 Interface in MicroStation Environment

Interfaces do not exist in isolation. Many interface functions and their performance are most effected by the environment which they are embedded. This project mainly uses MicroStation Development Language as its development environment.

One of the most useful features introduced with MicroStation is the OSF/Motif -- compliant graphical user interface. This graphical user interface includes (Intergraph,MDL Manual,1991):

(1) Tool palettes

(2) Pull - down menus

(3) Setting boxes and dialog boxes

This interface is controlled by the Dialog Manager, to which MicroStation Development Language has direct access. The MicroStation Development Language provides the designer with procedures to implement an interactive graphical user interface for the application program. It includes all the functions and tools needed for development of the user interface.

3.2.1 Dialog Manager

The MicroStation Dialog Box Manager controls the operation of the resizable MicroStation windows that appear on the screen. There are two types of windows, view windows and dialog box windows. View windows are used to display and change a MicroStation design file, while dialog box windows accept and process user input. Dialog boxes contain dialog items, which are user interface controls and are on-screen graphical entities which the user manipulates to change an application's variables. All dialog boxes in MicroStation are implemented using the MicroStation Dialog Box Manager. The Dialog Box Manager contains all of the features and capabilities required to implement a sophisticated graphical user interface application (Intergraph, MDL Manual, 1991).

3.2.2 Dialog Items

The actual components of a dialog box are referred to as items. The MicroStation dialog box manger supports standard dialog box items which are becoming the common features of a graphical user interface. It provides the user interface programmer with specially designed routines that handle these standard items. These tools can be selected to match the requirement of the interface's tasks. The advantage is that users have extensive control and great flexibility in creating the interface, while the disadvantage is that it requires a long time to gain proficiency.

The standard dialog items include the: Label, Group Box, Toggle Button, Push Button, Option Button, Scroll Bar, Text, Colour Picker, Level Map, Menu Bar, Text Pull-Down, Option Pull-Down, Tool Palettes, List Box, Generic etc.. Each item has its own function, so that when designing an interface application, the dialog item type should be matched to the intended operation. The standard dialog items and their main functions are listed in Appendix B.

The dialog manager breaks the dialog items into the dialog item resource and dialog item list specification. The dialog manager defines a dialog box by using these two specifications.

3.3 Designing the User Interface using MicroStation Development Language

3.3.1 Task of the Interface in Engineering Design

To establish mechanisms by which the designer is easily able to communicate with computer aided design systems, "the conventional tools of design have to be selected which are inherent to the design of a special product, or to the designer's activity in the design process. In addition, from the choice of devices and techniques for communication the ones which could aid and extend the designer's activity should be included" (Zsofia Ruttkay, 1987, p77). According to Zsofia Ruttkay, "there are well-established traditional ways -- textual description, sketches with symbols for parts, annotated drawing -- to present aspects of design", using different media, graphics, text and data sheets. "These traditional presentations should be supported when using CAD systems. They could be used not only to give information about a complete design, but to change and improve the design interactively during the course of the design process" (Zsofia Ruttkay,1987,p78).

In general, during the course of the design process, the designer, aided by the system, implements the design process by defining the value of any design attribute, and by initiating a design action, which is then executed by the program. When performing the design action, the information of the design attributes and the design's current state should be conveyed to the designer.

The conceptual basis for the user interface which is to meet basic and common requirements of engineering CAD design, concerning communication with the user, is:

(1) It should mediate between the user and the program: "to transform the user's inputs to appropriate communication action, and to generate the necessary output actions to inform the user according to the system's actions" and to indicate what input actions should be carried out next(Zsofia Rurrkay,1987, p81).

(2) It should also communicate at any stage what actions are allowed by the user and the program.

The interface between the system and user defined in this study is user driven, the user has much freedom in choosing what to do next and how to improve the design.

3.3.2 Architecture of the Interface

The interface consists of three main dialog boxes: the Input Information Box, Shaft Information Box, and Shaft Feature Selection Box as well as nine sub-dialog boxes. Each one is specialised in a certain aspect of the overall design task. The Input Information Box starts the design process with the design specification. It accepts shaft basic force and torsion data. The Shaft Information Box performs the shaft preliminary design. It collects bearing information and initiates other design actions, such as shaft layout, calculation of the basic shaft diameter and shaft detail design. The Shaft Feature Selection Box carries through the detail design. It provides a facility to design a shaft in the form of features. There are other sub-dialog boxes to attach to the main boxes. The main three boxes are shown in several figures below. The functionalities and features of the interface, and techniques to create the

graphical interface with MDL are highlighted throughout the rest of this section.

The dialog boxes are linked together forming the user interface from which the whole design procedure can be conducted by following the design steps invoked. It is worth mentioning that these dialog boxes which construct the user interface are modeless dialog boxes. This allows the user to interface with more than one dialog box at any given time. There are "live" boxes waiting for mouse events. This enables the designer to easily examine and manipulate the design information and provides the flexibility to navigate through the system and perform the desired task in an efficient way.

Input Information Box (Fig.3.1) - The main function of the Input Information Box is to start the design session by receiving input data. It uses a "form fill-in" format as its main interface style. Form filling is the most commonly used dialog type for data entry, retrieval and editing. It is done by displaying a form on the screen, this display has a title, labels for the variable area and markers to show where the data should be entered. Sometimes forms may include a message area for instruction and explanation. The cursor is software controlled to move from one field to the next by using the Tab or Carriage return key. This can also be done using the mouse. Data can be retrieved, displayed and edited after entry by using the same method.

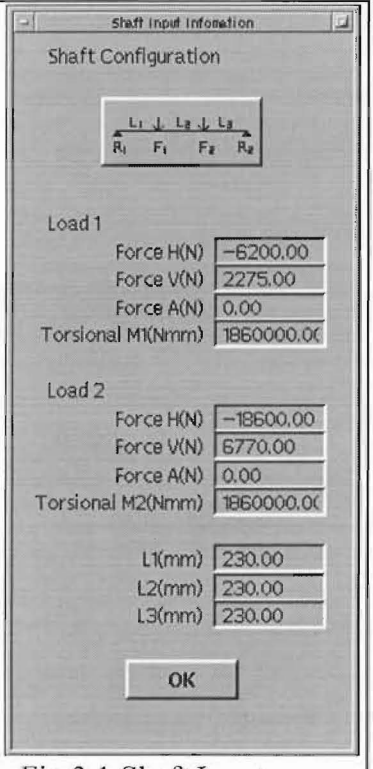


Fig.3.1 Shaft Input Information Box

The standard items used in this dialog box are the Text, Label, OptionButton and PushButton. The Text part of the Input Information Box will prompt for the shaft forces which are divided into components in the coordinate direction. The Label gives the meaning of the graphical icon. The OptionButton shows shaft configuration forms and the PushButton will initiate the functions. The designer's input data is stored as editable strings. They can be a single character, any string, integers, entry data floating point numbers, or working units strings. The entry data format information is defined in the Text Item resource specification. The permissible values can also be specified if required. The text item currently processing designer keystrokes is in focus. When a data entry area is ready to accept keystrokes, it will darken. Horizontal scrolling allows the editing of more characters than would otherwise fit inside the text item. Multiple characters can be selected by dragging the mouse cursor through ranges of text (Intergraph, MDL Manual,1991).

Normally, there are some design variables in the application which should be bound to required data by the designer in the design process. MDL provides the facility to perform this function through the graphical user interface. Among the information that can be included in an item resource specification is a string that determines which application variable the item controls. The application variable underlying a particular item is specified in the access string field of the item resource specification. Access strings define variables, structures or pointers to structures, for the dialog item to inspect and modify. The dialog manager uses C expressions to handle access strings. Eg. "Force and Torsion" structure in the header file (shaft.h, The MDL file type is

illustrated in Appendix C) is created to hold the variables of forces and torsions in shaft design that the text item can manipulate. The text items in the Input Information Box control the members of variables of this structure. Also the permission for dialog items to manipulate the input data must be given by "publishing the variables". That is the mdlDialog_publishComplexVariable function must be called in the source file (shaft.mc) to publish the structure type variable to the MDL C expression handling functions. In addition, because the variable is an instance of structure, the publish structure statement in type declaration resource file (shaft.mt) is required. eg. publishStructures (forceNtorsion). This statement identifies the structure to be defined in the resource file. In this way, variables in the application are connected with the dialog item and therefore can be bound to the required data by the designer through the dialog box.

Another feature of this dialog box is the use of icons from the option choices, as shown in Fig.3.1. Normally, an OptionButton item lets users select a single choice from a small group of options. When pressing the mouse button while the cursor is in the current choice, the list of available choices appear. Dragging the cursor highlights the choice. Releasing it makes the highlighted item the current selection. The option choice can be a text string or icon. In this study, the various combinations and locations of a shaft and its force and torsion elements are listed as icons. The designer can chose the icon which matches the design configuration. Icons are defined in OptionButton resource and the icon resources specifications in resource file (shaft.r). The detail of the definition process is listed in Appendix D.

Shaft Information Box (Fig.3.2) -- This interface consists of all the necessary functions required to assist the designer in the preliminary design stage. It computes the bearing reactions acting on the shaft and collects the bearings basic data other than bearing load. It activates the design action for a shaft layout drawing and calculation of shaft diameter based on failure under peak load. Finally a drawing of the shaft geometry is activated using the calculated basic shaft diameter.

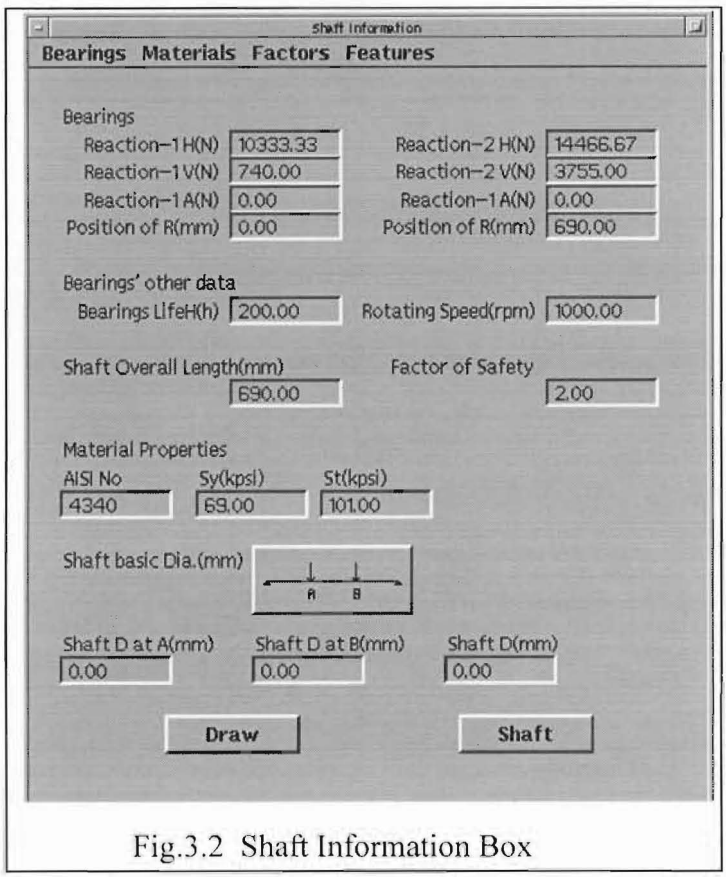


Fig.3.2 Shaft Information Box

In this dialog box, lots of Text items are used for the design data, such as bearing information data, material properties data and safety factors. Some of this data is generated during the design process rather than from designer input. For this sort of data, the problem is how to get it back to the designer via displaying it in the dialog box. The state of a dialog item is composed of two parts: internal value versus external state. The internal value of a dialog item is used to determine the item's appearance, while the external state is the value of data that is specified with access strings. When the designer finishes the Input Information Box and pushes the 'OK' PushButton, the values of the bearing reactions are calculated and bound to the variables of reactions

through access strings, which relate with the bearing reaction text item. In order to make the bearing reaction Text item display the data that item presents, the item's internal value is forced to match item's external state by calling function: mdlDialog_itemSynch in the source file (shaft.mc), that is the items are synchronised. In this way, the calculated bearing reaction data is displayed in the Shaft Information Box (Fig.3.2). An example is given in Appendix D.

In certain situations, a change in the external state of one item requires the simultaneous change in the appearance of a number of other items and sometimes these simultaneous changes come across the different dialog boxes, eg. the status of the shaft configuration option item in the Shaft Input Information Box needs to be shown as another item in the Shaft Information Box. This simultaneous display can be accomplished with the use of the synonym resource in the resource file (shaft.r). The two OptionButton items in the two boxes are associated with the following synonym resource in shaft.r file:

```
DItem_SynonymsRes  SYNONYMID_ Confg=
{
  {optionButton,OPTIONBUTTONID_Configuration1}
  {optionButton,OPTIONBUTONID_Config1}
}
```

Therefore, when one of the configurations in the Shaft Input Information Box is chosen, a synchronised message is sent to the OptionButton in the Shaft Information Box. The appearance of the OptionButton in the Shaft Information Box automatically matches the state of the item in the Shaft Input

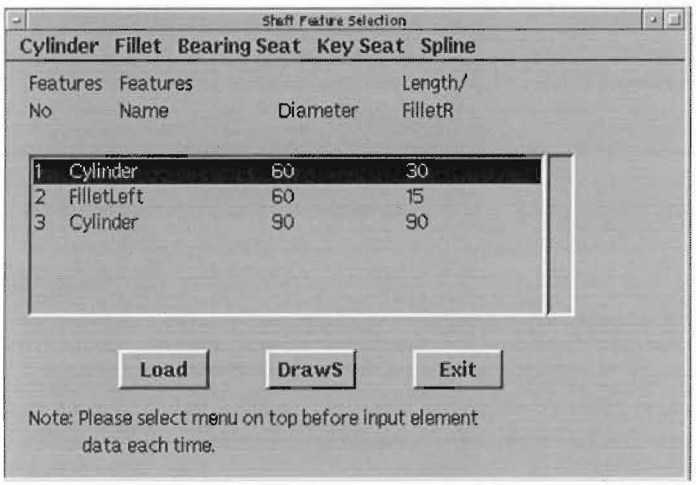
Information Box.

Other than Text items, the Menu Bar has been designed to hold a number of pull-down menus. These pull-down menus appear on the screen in response to a click with a pointing device - eg. mouse. Selection can be made by moving the mouse over the menu items. These pull-down menus are either Text Pull-Down menus or Option Pull-Down menus. The Text Pull-Down menu contains items that display as text strings and they are used to activate a design action -- for example the process of choosing bearings, choosing materials and shaft detail design (Fig.3.2). This can be done by attaching a command task ID in the pull-down menu resource specification in the resource file (shaft.r). Selecting an action from the Text Pull-Down menu invokes execution from the user functions or commands. The commands related to texts ('Bearings', 'Materials', 'Features') in the menu initiate Bearing, Material and Feature Selection dialog boxes respectively according to the step of the design procedure. The Option Pull-Down menu gives several safety factor choices from which designers may choose (Fig.3.2). The value of the safely factor which is selected goes to the variables through an access string defined in the Option Pull-Down menu item resource specification in the shaft resource file.

Another way to process the design action is using PushButtons via a dialog hook which is one of the sub-systems through which the dialog manager handles dialog boxes. A dialog hook is a function designated by the application to be called when the item associated hook is manipulated. The hook function gives the MDL dialog box programmers the most power and flexibility in modifying or amplifying the default behaviour of the standard

dialog box item as a result of user actions within dialog boxes. A hook function can be attached to a dialog box, in which case it is a dialog hook function, or to an individual dialog item, in which case it is an item hook function. The details of the process are explained in Appendix D. In this box, two hook functions are attached to PushButton 'Draw' and 'Shaft' which initiate design actions, drawing a shaft layout and calculating basic shaft diameter, respectively.

Shaft Feature Selection Box (Fig.3.3) -- This box is associated with other five sub-dialog boxes and interactively gathers information about design attributes of the shaft. It enables designers to create and edit any shaft features which are identified from the pull-down text menu. This opens the related feature box through which the current feature attributes value can be manipulated or changed. Once the desired data has been input , it can be transferred to the Feature Selection Box. All of the shaft feature data is displayed in the Feature Selection Box as text string lists. The designer then has the chance to study this design and edit any part of it when required (Fig.3.3). More details of these interfaces are described in Chapter 5.



The main item used in the Shaft Feature Selection Box is the List Box which has been designed to display, manipulate and control the text string list. The string list manager which is a function embedded in MDL also aids in the

use of strings. These strings can be contained in a resource file by defining a stringlist resource in the file. The string data can then be obtained and stored by calling the appropriate string list manager functions via the dialog hook. More details of the functions and techniques related with control of stringlists are introduced in Chapter 4 and Chapter 5.

Other than the interface described above, there is another way for the user to get messages conveyed by the program. MicroStation's command window has six areas where messages of different type, such as error, prompt, command, status, and so on, can be displayed by calling the mdlOutput_ functions. Also, messages can be pre-defined in the resource file as MessageList. These are useful functions for the user to get the status and information of design activities from program.

Chapter 4 Database Management

Lots of data is either used or produced during the engineering design process. Sakthivel and Kalyanaraman (1993) classify this data into two groups: static data and dynamic data. "Static data is the common handbook type of data on materials, equipment and finance used in the engineering process, whereas dynamic data is the problem context-dependent data, generated during the actual engineering process" (T.S. Sakthivel and V. Kalyanaraman, 1993, p5). CAD systems require extensive and efficient data management capabilities to handle this static and dynamic data which needs to be stored, retrieved, manipulated, and updated during all the phases of design process. Many CAD systems employ databases to store and describe engineering data. Using a database to manage data has many advantages. Some of these advantages include: the database can store and access data independent of its use, so that the data can be called in the design process whenever required; the relationships among the data can be stored in the database, so that the dependencies are documented; and databases manage data with consistency and integrity, so that data can be manipulated easily.

In this chapter, static data management in a design process for the MicroStation environment is discussed. The management of some dynamic data is described in Chapter 5.

4.1 Structure of Database

Zeid (1990) defines a database as "an organised collection of graphics and non graphics data stored on secondary storage in the computer. It could, therefore, be viewed as the art of storing or the implementation of data structure within the computer" (Ibrahim Zeid, 1991, p90). The most basic data structures are records and fields. A record can be simply defined as a collection of fields or attributes while fields are the elementary data items.

In this study, engineering static data are stored as a table composed of rows (records) and columns (fields). Each row corresponds to an individual object, eg. one kind of material, and the properties of objects are represented by columns. The database is created and stored as an ASCII file which can be accessed sequentially, so that a database can be edited and updated with new information simply using the DOS editor.

The database described above is used to handle static engineering data, such as material properties and engineering standard parts like standard keys and these data are grouped into different directly accessed ASCII files according to their type and characteristic. The material database contains information pertaining to the mechanical properties of different materials. Properties such as the processing method, yield strength, and AISI numbers are defined as attributes while different materials are listed in rows. Table 4.1 shows an example of the material database. The standard engineering parts database, such as standard keys, contains the geometry data of the each part (Table 4.2).

UNS Number	AISI Number	Processing	Yield Strength	Tensile Strength
			kpsi	kpsi
G10100	1010	HR	26	47
G10100	1010	CD	44	53
G10150	1015	HR	27	50
G10150	1015	CD	47	56
G10180	1018	HR	32	58
G10180	1018	CD	54	64

Table 4.1 A Sample Database of Materials

Shaft	Diameter	Key	Section
over (mm)	incl (mm)	width (mm)	thickness (mm)
6	8	2	2
8	10	3	3
10	12	4	4
12	17	5	5
22	30	8	7

Table 4.2 An Example of Standard Parallel Key Database

4.2 Interface of Database

In this case the main task of the interface is to allow access to the data stored in the database. It links the database to the design modules allowing retrieval of information from the database while the design takes place. The interface forms a layer of software between the database itself and the user of this database (design variables). In another words, it is a software function that acts as a liaison between the database and the design variables (Fig. 4.1).

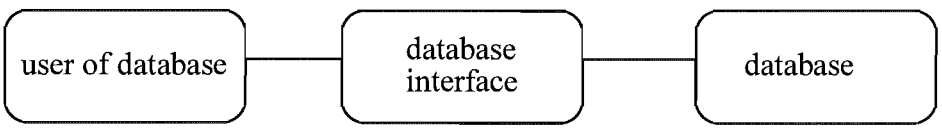


Fig. 4.1 Data Flow for Intergration with Database

The interface consists of several list boxes for display of the database, such as the Material List Box, Key List Box etc. This interface provides two main functions:

(1) allows access to the data in the database, and

(2) selection of records from a List Box and binding data to design variables.

A sample of the interface for material selection is shown in Fig. 4.2.

List Box items allow the display and selection of multiple text strings. A List Box has rows and columns which match the records and fields in the database. This item is used to create, manipulate and destroy string lists. The information in the database is retrieved into a List Box as a string list. Each attribute in the database is set into the string list as a stringlist member and the List Box item assumes that there is one string list member per column. The List Box must have an item hook function attached to it. The Dialog Manager communicates with the hook function via the Dialog Message Structure (Appendix E). String lists creation, manipulation and destruction are performed based on the message which the hook function receives.

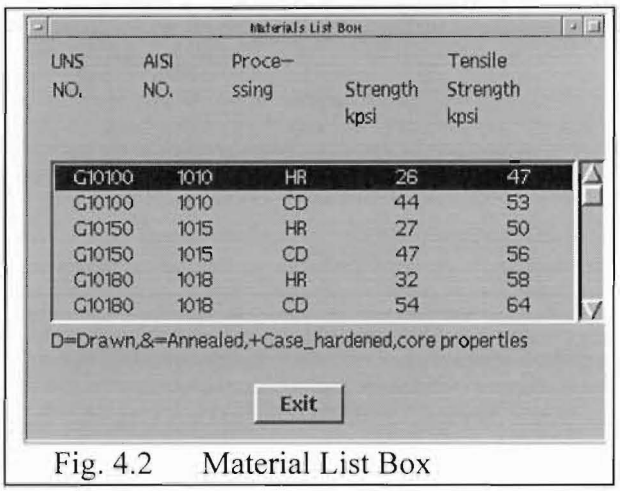


Fig. 4.2 Material List Box

The retrieval function is illustrated in Fig. 4.3. When the list box hook function receives the DITEM_MESSAGE_CREATE message, it connects the string list, which is created by calling mdlStringlist_create, to the List Box

item by calling mdlDialog_listboxSetStrListP. The mdlDialog_fileOpen function enables the user to open a desired file from one of several options. The data in the file are read into a buffer and separated into each attribute by tokenizing them with spaces and commas. The mdlStringList_setMember then sets the member to each attribute of database. The program will continue to read data file and check if there are any data left until the pointer reach the end of the file. The data are displayed in the List Box. A List Box has a vertical scroll bar that is used to scroll through the list of text strings when there are more strings than can be displayed within the List Box item at one time.

When the designer uses a mouse to go through the content in the List Box, the current row is darkened. If the designer clicks the button, the DITEM_MESSAGE_STATECHANGED message is sent to a list box hook function. The mdlDialog_listBoxIsCellSelected function then checks which row in the List Box is selected. Getting the string list which is currently connected to the List Box is done by calling mdlDialog_listBoxGetStrListP. The mdlStringList_getMember sends data from the List Box to the buffer and then the data can be transferred to the design variable by reading data from the buffer. The flow chart about this process is shown in Fig. 4.4.

Finally, the string list that the List Box is manipulating will be destroyed when the hook function receives the message DITEM_MESSAGE_DESTROY.

4.3 Discussion

The database management discussed here is used to explain the simple and basic approach to store and use the data. It focuses more on investigating the MDL environment rather than details of the database management system itself. In fact MicroStation provides a channel to connect to Oracle or dBASE, which are the more common non-graphic database software packages used. Also development of the intelligence design systems requires an intelligent database system to handle the large amounts of data generated during the design process. This is beyond the scope of this study.

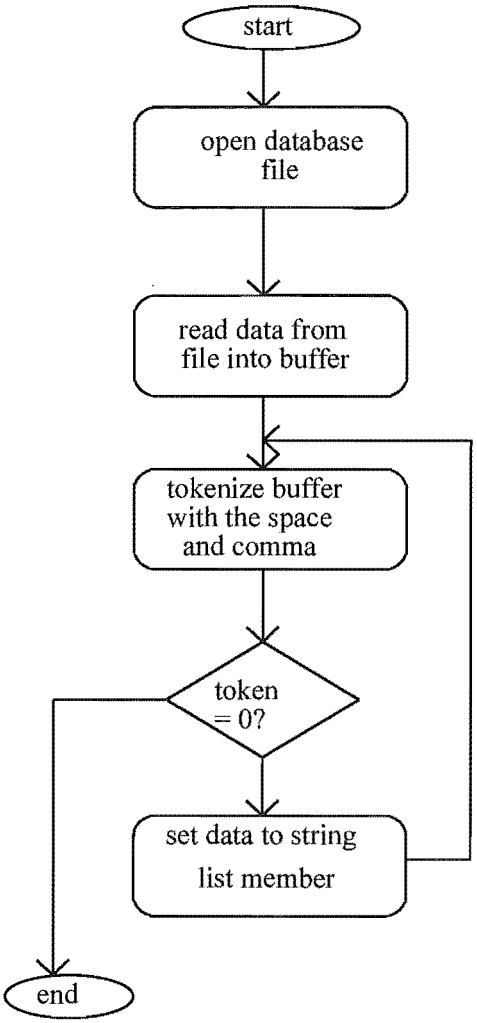


Fig.4.3 Flow Chart for Retrieval Function Module

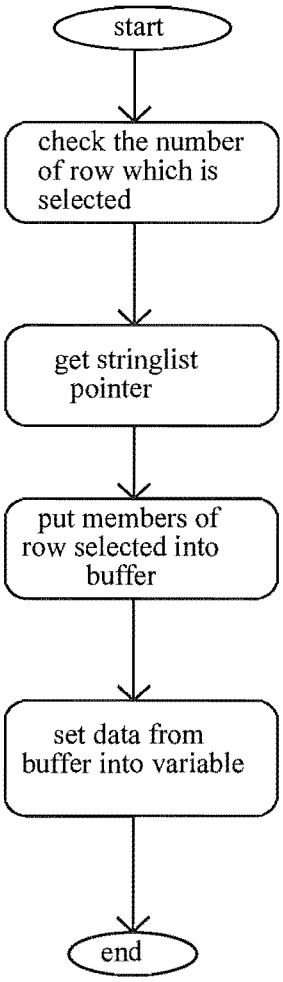


Fig.4.4 Flow Chart for Selection Function Module

Chapter 5 Feature Based Design

The concept of features was first introduced in seeking a connection between design and manufacturing. Feature technology has spread in the field of design, analysis, process planning and manufacturing, etc. Design by features is a relatively newly established technique and is used increasingly in computer aided design and computer aided manufacturing. The idea of design by features is simply designing the engineering components in terms of features. This design method has become the "most promising approach to enable computers to comprehend mechanical objects in computer-aided design" (Schaal and Ehrlenspiel, 1993, p77). In this section, feature based approaches in detailed design phases and the feature description concept in the MicroStation environment are described.

5.1 Concept of Feature Based Approach

The concept of a "feature" is being used for many different purposes. The very general definition is as follows:

"A feature is a partial form or a product character that is considered as a unit and that has a semantic meaning in design, process planning, manufacture, cost estimation or other engineering disciplines" (Wierda, 1991, p3).

That is, "a feature is a set of information related to a part's description. The description could be for design, or manufacturing and inspection or even for administrative purposes" (Shah and Rogers, 1988, p7).

When considering the design process, a feature is considered as a "design feature, in terms of its geometry, specifications and details to fulfil certain functional requirements" (Case and Gao, 1993, p3). In some research, features are considered as design-objects, which are regarded as individual components or parts. Features based on this concept can be organised into a hierarchy of classes.

Applying these features concepts to shaft design, the shaft can be decomposed into its basic features, such as bearing seat, cylinder, fillet, key seat and spline. These basic features are defined as low class. The whole shaft itself is a single feature in an assembly plan, which is defined as a higher class. The discussions here are based on basic shaft features.

5.2 Feature-based Detail Design Module

The shaft detail design process can be defined as a shaft "basic features" approach. This section provides insight as to how the feature-based detail design module was developed. The architecture of the procedure is illustrated in Fig. 5.1.

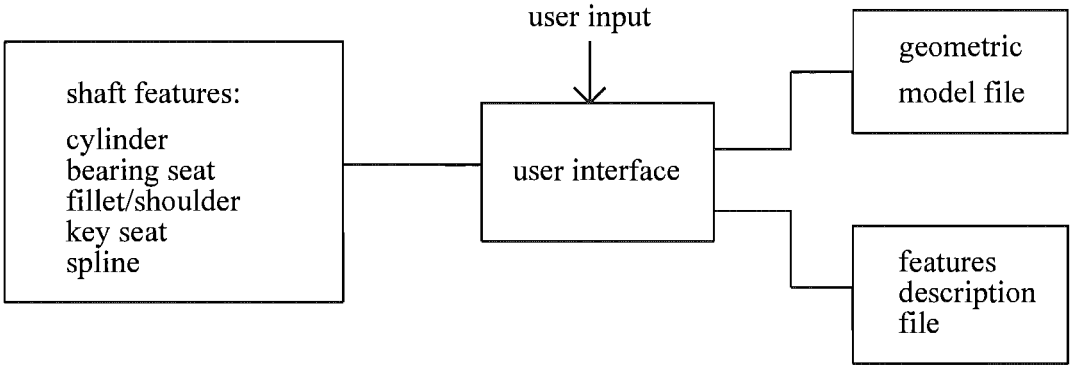


Fig.5.1 Architecture of Feature-based Approach

From the preliminary shaft design result, which is the basic shaft diameter, the next step is to design a shaft in the detail phase. Through the user interface, the designer can select from a predefined set of shaft features and add appropriate parameters to form a shaft, which then can be examined and appraised. Data control and drawing of the shaft are performed via the embodied functions within the program.

As the result of this approach, two files are created. One is a shaft features description file, the other is shaft geometry file. Of these two files, one is output as MicroStation geometry data file with MicroStation data format (.dgn) , the other as a text string file which includes the features description and is stored in an MDL resource file.

The module's flow chart is shown in Fig.5.2.

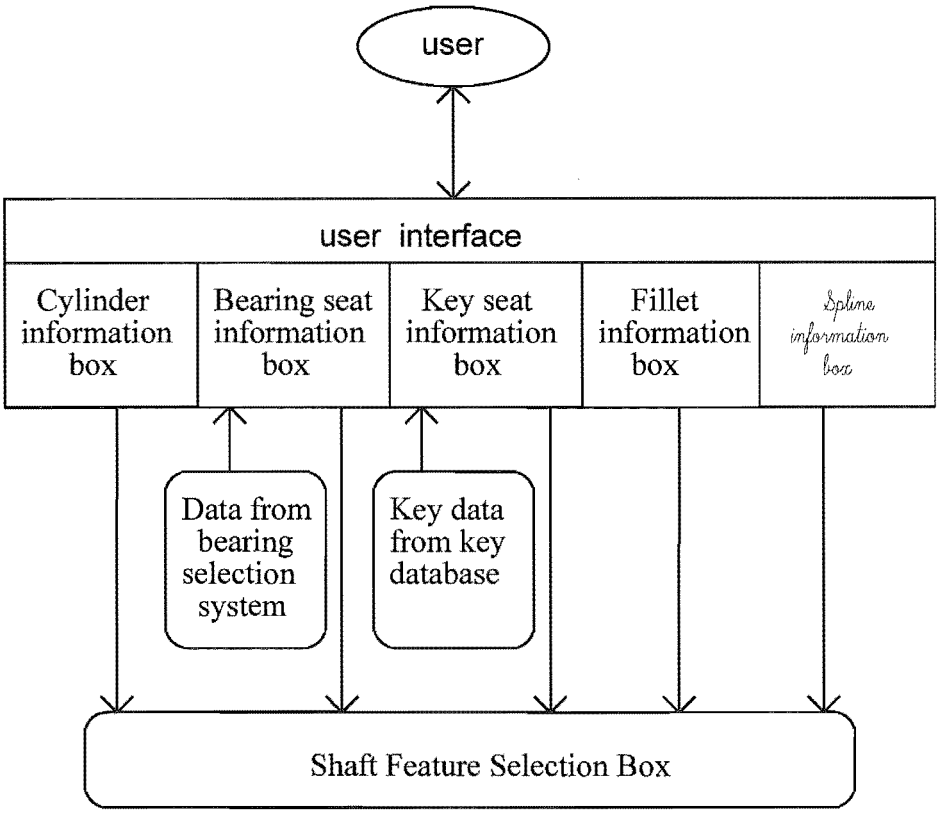


Fig.5.2 Flow Chart for Shaft Feature Selection Module

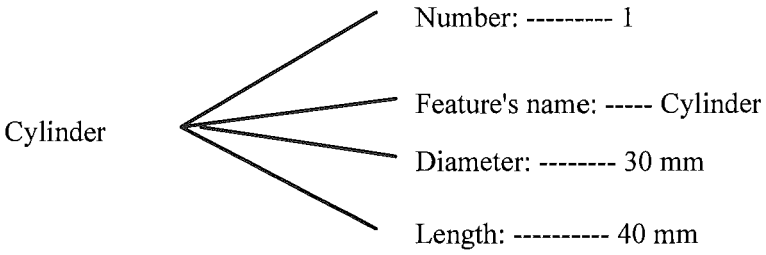
This module is capable of :

1. designing features and setting the shaft features data into stringlists displayed in the List Box;
2. saving the shaft features description into a stringlist resource file;
3. loading the shaft features description and modifying any part as required;
4. passing the whole features description to the geometry modelling module;

5.2.1 Data Structure

The concept of a feature defined earlier in 5.1 strongly resembles the notion of an "object" as defined in artificial intelligence. Because of the

similarity, the methods used in artificial intelligence to describe objects are adopted to features. Frame representations, which can be conceived as property lists of objects, are well suited to structure and store features data (Schaal and Ehrlenspiel, 1993, p72). A frame, typically consisting of several fields, is a data structure which represents a feature description. Each field can be related to a feature attribute. The attributes are composed by the feature number, the feature name and feature's parameters. The feature number corresponds to the position of the feature along the shaft. So that a frame is tied directly to a specific feature. An example of feature frame is shown as follows:



In this way, the whole shaft can be described in the form of a group of object (feature) descriptions. This forms a database which collects the features geometry data and their relationship to each other, which is defined in an assembly order (Fig.5.3). The database is generated during the design process as the designer's input and can be stored, edited and updated as a file.

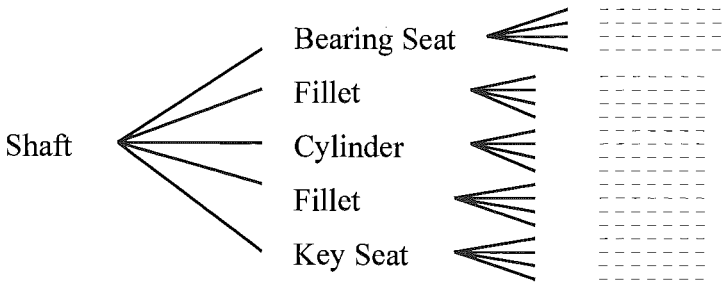


Fig.5.3 An Example of Shaft Database

5.2.2 User Interface

As descried in Chapter 3, the task of this interface is to control the feature-based approach in the detail design stage. The requirements for the interface are as follows:

(1) access to a feature description box;

(2) access to a engineering standard part database;

(3) access to functions, such as 'draw a shaft' and 'load shaft data';

A feature menu within the dialog box provides access to any of the feature input boxes. Selection of a feature name, eg, cylinder, will open a feature description input dialog box, which is a data entry form. Then, user can manipulate the feature parameters by using this on-screen entry form. Fig. 5.4 shows a sample of a cylinder entry form. Each form corresponds to a specific feature and contains numerous prompts which describe the feature parameters. The data entered on each form corresponds to a single row in the Feature Selection Box, each prompt, in turn, corresponds to a particular field of the row. Data is entered into the form by typing a response to each prompt. Once the data entry into the form is complete and determined to be accurate, the user can select the 'OK' button, the data is then transferred to the Feature Selection Box as a new row in List Box item. As a result of this input, the database for shaft information is formed.

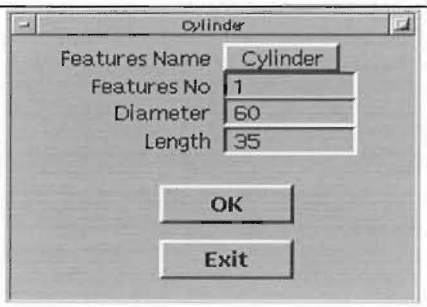


Fig.5.4 Cylinder Input Box

The resulting database which is described in 5.2.1 will be present in the List Box. The designer can review the descriptions and edit them if necessary. The feature description data box can be used in a similar manner to modify the data. In the specified feature description box, the user simply types over the new value for each prompt then select the 'OK' action in the box. The old value is then overwritten by the new one.

The feature's geometric description may depend on the properties of specific engineering parts, eg. the dimensions of a bearing seat depend on the length and diameter of the bearing which it will support and the data of a key seat will be related with the parameters of standard keys. When the designer inputs the feature's data, the relevant information should be presented and set to the related attributers of the feature. For example, when defining a bearing seat, the bearing selection box will be opened along with the bearing seat box. The bearing data will be saved into a file after executing the bearing selection system which will be discussed in Chapter 7. The 'read' command in bearing selection box will launch the function which will read the bearing's data from the file and set the relevant data into the bearing seat length and diameter items in the bearing seat box.

Other than accepting input from the designer, this interface provides access to functions embodied within the program, such as data and information control as well as the drawing of a shaft.

5.2.3 Data Control Techniques

As discussed above, feature descriptions defined by the designer are stored in the form of string lists. These string lists can be created, updated, modified, deleted and displayed in the List Box. The data can also be saved into and loaded from the resource file. The key to achieving these operations is to use the string list manager which is an MDL embodied function and a pointer attached to the List Box and sting list itself, named string list pointer.

The program functions mentioned above are illustrated as follows.

Before the designer's input feature data, space must be allocated for the string list and a pointer is declared to the string list. The program then link the List Box with the string list by attaching the string list pointer to the box. Finally, because the string list data will be manipulated from the designer's interface, the string list pointer is stored as user's data in the "user data" location which MicroStation allocates. The string list pointer can be called when transferring input data to a string list (Fig.5.5).

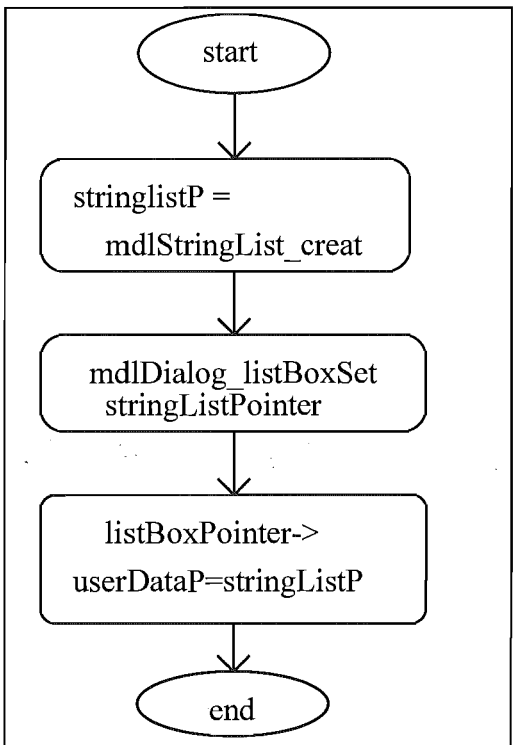


Fig.5.5 Flow Chart for Creating Stringlists

When the user has completed the feature input and presses 'OK', the function is activated via the dialog hook. The function starts by retrieving the Shaft Feature Selection Box and the List Box item in the box. The program

then gets the string list pointer which is declared previously from the location which MicroStation allocates for the user data. Because the input data can be new or changed data, it is necessary for the program to check the data. If the data are new they are put into string lists straightaway. If not, the program deletes the old strings first then inset the new one. Finally, the program informs the List Box if the number of rows in its string list has changed and draws the new content in the List Box. Fig.5.6 shows the steps of the function.

'Save' and 'Load' functions have been developed for the operations associated with feature description data as string lists. The flow charts for these two facilities are shown in Fig.5.7 and Fig.5.8 respectively. The functions are activated by dialog hooks attached to the 'Save' and 'Load' buttons.

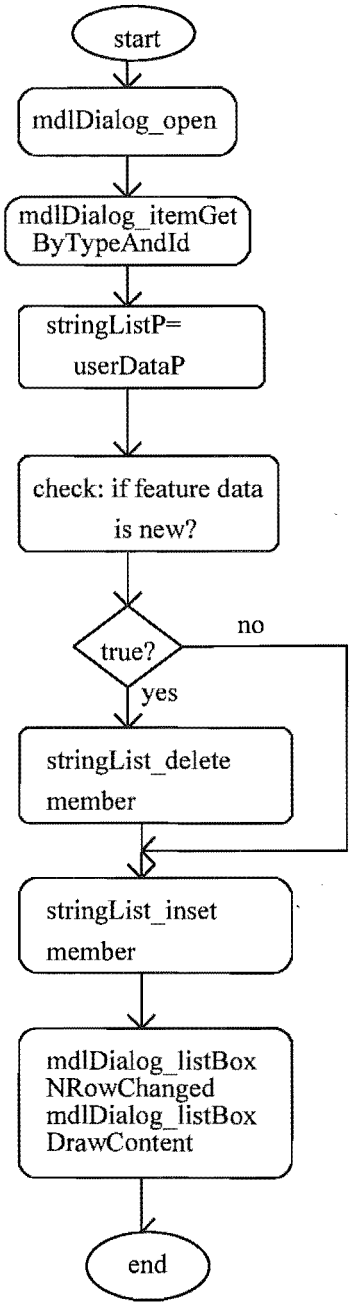


Fig.5.6 Chart Flow for Putting Strightlist into the Shaft Features Selection Dialog Box

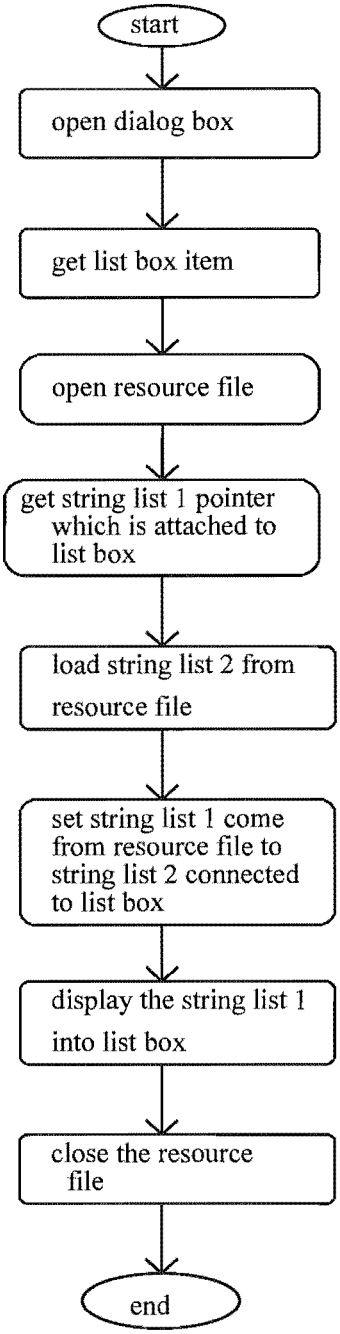


Fig.5.7 Flow Chart for Load Function

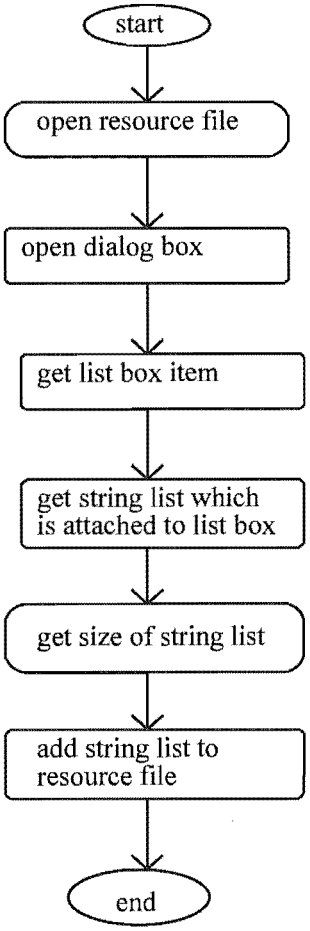


Fig.5.8 Flow Chart for Save Function

5.3 Geometric Modelling

Geometric modelling is one of the computer aided design tools. The CAD systems have been developed around various types of geometric modeller to describe the shape of engineering objects. Modellers are typically

of three types: wireframes, surfaces and solids (Ibrahim Zeid,1992,p153). MicroStation has wireframe and surface modelling capacity. With the features based approach in engineering detail design, a feature is used as a basic design object. Each feature should be resolved into a description based in primitive geometric modeller elements. In this study, the feature geometric description has been constructed by sweeping a surface representative along a guideline and the shaft geometric descriptions is then described by a collection of geometric description of appropriately positioned features. The shaft draw module structure is presented in Fig.5.9.

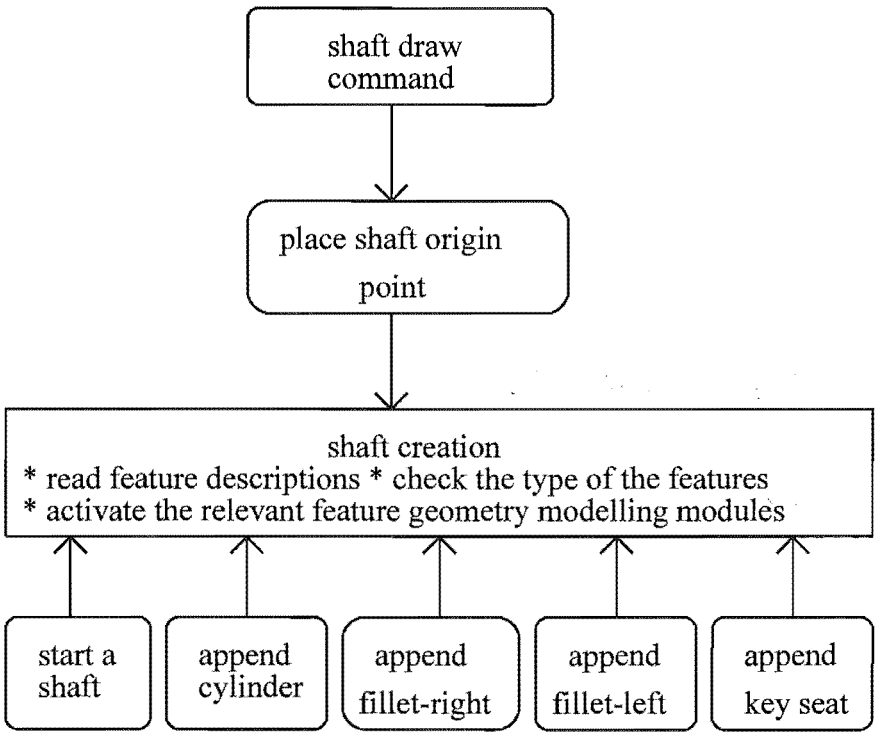


Fig.5.9 Flow Chart for Shaft Draw Module

The parameters for the description of certain features are accepted by the program through the user interface box and the feature data generates a feature geometric description file from which the full geometric models can be accessed. The draw action is effected by the shaft draw command via the

dialog hook which is attached to the 'drawS' PushButton. The program will ask the designer to place an origin point for the shaft in the MicroStation view windows. The program reads the feature description file and checks each feature type. Then the program calls relevant feature geometric modelling models and creates a shaft by appending these shaft feature models starting from the left-most feature and working towards the right-hand side.

The strategies used in geometric modelling are as follows:

(1) The feature geometric model is defined by the geometry primitives provided by MicroStation. eg. lines, arcs, shapes etc. which correspond to the MDL element creation functions mdlArc_create, mdlShape_create, and mdlLine_create etc. These elements can then be grouped into a complex chain header element by calling mdlComplexChain_createHeader. These MDL facilities are used to create a specific 2D surface shape.

(2) A solid may be defined by a 2D surface shape rotated around an axis. This can be done by the mdlSurface_revolve function call (Fig.5.10).

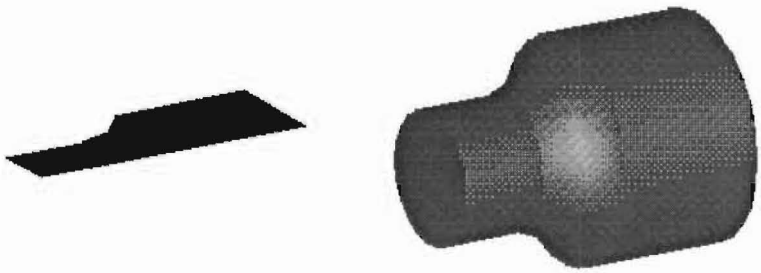


Fig.5.10 Solid Formed by the Rotation of a 2D Surface

(3) In MicroStation, elements are stored sequentially, in design files. Each primitive element has a fixed format header that contains the element's size so that the next element can be found. MDL programs can create or manipulate elements by declaring a buffer. However, complex elements are stored in the design file with a complex header element followed by a series of component elements. The component elements can not be modified without information in the header element being updated. Since the number and size of component elements can vary, MDL programs are unlikely to declare a buffer large enough to hold complex elements. MDL provides a series of functions, known as element descriptors. These element descriptor functions enable components of complex elements to be manipulated without the need to maintain the headers.

In this study, the element descriptors are used to compose a linked list of elements to form the geometric description for the features. The follow steps should be taken:

(a) create a complex chain header element;
(b) create a new descriptor from an existing element;
(c) create a simple element;
(d) append element to the descriptor;
(e) repeat step (c) and (d) until all simple elements are created;
(f) display descriptor in all views;
(g) add all elements in the descriptor to the file;
(h) free memory used in element descriptor chain.

(4) As mentioned above, the shaft will be drawn as an assembly of features. Each feature geometry model is given its own absolute location in global coordinate space. The shaft is an assembly of features, so that each feature is positioned and connected to each other in the assembly model. The operation is controlled by MDL current transformation functions.

Before the drawing action starts, the program first sets the current transform to correspond to the shaft's coordinate system. The copy of current transformation, which is a transformation matrix that transforms coordinates or distances between MDL applications and MicroStation, is pushed onto a stack (Intergraph,1991,MDL Manual). The program will translate the origin of the current transformation to the given starting point selected by the designer. As the program models each feature, it moves the current transformation matrix's origin to align it with the feature's axes (Fig.5.11). When the whole assembly, that is the shaft, is finished, the program will remove all transformation matrix from the stack.

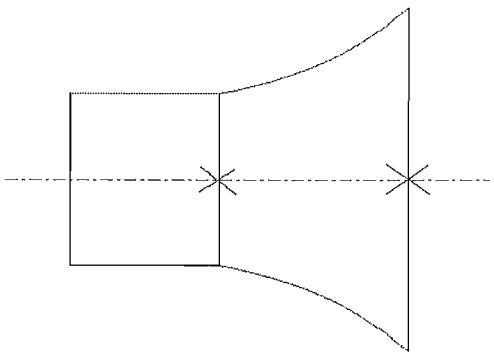


Fig.5.11 Transferring the Origin

According to this approach to the assembly model, there is a definite order. One feature is fixed and each of the subsequent features is positioned one at a time, relative to those whose position have already been fixed. This provides a natural approach for an assembly sequence.

5.4 Discussion

The term 'feature' used here is more like a geometric object. However, the pure geometric modelling definition does not include the reason for a feature's existence or usefulness. Because features encode the engineering significance of the geometry, the definition should include the purpose for which a feature is useful. It could be a geometric object with additional non-geometric properties. The feature's description should include the information related to the manufacturing. This is something which must be addressed in any fully developed program if it is to be of maximum usefulness.

Chapter 6 Program Communication

Engineering design incorporating computers has resulted in a significant increase in the amount of engineering information which can be generated and manipulated in the design process. Many types of design software programs treat specific aspects in the design process, so that design software programs involved in a common design need to exchange information to each other in a consistent and efficient manner so as to enable the full collaboration of programs in the design process. Interfaces are used for passing the required information between the programs. The medium and compatibility of data structures that facilitate the transfer of information from one design software program to another is essential for effective communication. This chapter investigates data exchange in the local environment and the methods of communication other programs use to interface with MicroStation.

6.1 Data Transference Between the Software

The use of CAD/CAM has grown extensively in recent years. Many different design software packages such as CAD systems, analysis programs, knowledge based design systems, CAM systems, etc. have been produced and implemented. The need to exchange data between these packages is "directly motivated by the need to integrate and automate the design and manufacturing process to obtain the maximum benefits from the CAD/CAM systems"

(Ibrahim Zeid, 1991, p443). There is always the demand to be able to share common data. Normally, the software packages have their own data management systems set up within the packages. Some programs use different data formats to represent the same thing, the definition, largely geometric, of an object (Peter F. Jones, 1992, p175). In some similar CAD/CAM systems the methods of data storage and structure are compatible and therefore information can be transferred directly. Design files in MicroStation and EMS are such a case. However, many dissimilar CAD/CAM systems store data in their own data structure, and there is no direct similarity between data entities. As a result problems often arise in data communication.

According to Jones and Zeid, two basic approaches are employed to solve communication problems. One is to directly transfer data from one CAD/CAM system to another via converters. Two converters are needed for each pair of systems as transfer may occur in either direction of a system pair. The other approach is to adopt an independent format, called a neutral format as the medium for data translation. This format acts as an intermediary and a focal point of communication among dissimilar data format CAD/CAM systems. Hence there is also a pair of converters for each system. A pre-processor transfers outgoing CAD/CAM data into the neutral format while a post-processor converts incoming data from the neutral format.

A number of neutral formats have been developed to meet the need to exchange CAD drawing files. In this study, the neutral formats needed for exchanging data in a local environment are investigated (Appendix G). Software packages have been grouped into languages, database and word processor, CAD and CAM. Appendix G gives a simple map which shows the

required data format when transferring the data between the packages. The main focus of the study was on MicroStation, Algor, and MasterCAM which represent computer aided drawing, finite element analysis and computer-aided manufacturing respectively and which cover the basic CAD/CAM system.

MicroStation Version 4 provides DXF (Drawing interchange Format) converters to import and export drawing files to and from different CAD packages. "The DXF format was originally a format devised by the suppliers of AutoCAD to write out a drawing description as ASCII text for processing by other software. As a neutral file format it is limited to the repertoire of entities found in AutoCAD" (Peter F. Jones, 1992, pp180-181). Conversion between DXF and MicroStation can not be exact because elements are defined differently. DXF format does not support some MicroStation elements, such as multi-line text, B-spline, true complex shapes and complex chains, etc. (Intergraph, MicroStation User's Guide). MicroStation Version 5 provides another principal neutral format, that is Initial Graphics Exchange Specification (IGES). IGES is an international standard, Version 5 of which is now current, and finding widespread use in CAD/CAM data exchange.

When finishing a shaft drawing in MicroStation, it needs to be transferred to Algor for analysis. Algor is a finite element analysis program which analyses stresses and deformation within a mechanical member. Algor accepts several data formats which are imported from other CAD/CAM packages. There is a converter for MicroStation which converts incoming data from DXF format. Because of the limitations of the entities supported by the DXF format, all of the shaft complex shapes must be dropped into simple

elements such as lines, arcs, etc. before converting to DXF format. Algor also has a converter to accept IGES format.

To favour meshing in finite element analysis (FEA), the appropriate geometry to be transformed to the FEA system must be considered. The finite element analysis process begins with the designer completely specifying the problem on a geometry basis and then the geometry models are meshed to create a finite element model. The attributes (loads, boundary conditions, material properties, etc.) are mapped from a geometry basis to a node and element basis. Finite element analysis is performed and the results interpreted. The meshing process is a basic step of finite element analysis and it is a model conversion process, the geometric model is transformed into a discrete model which is a finite element representation. The finite element representation consists of a "collection of cells which must satisfy a number of geometrical and topological conditions dictated by the finite element method" (Sapidis and Perucchio, 1992, p517). Algor provides several mesh generators, such as 2 object, 3 point, and 4 point.

In the case of shaft design, the basic geometry is a cylinder-like shape. In order to favour the meshing procedure a better way to construct the finite element mesh is to follow a three-step strategy:

step 1: transfer the shaft geometry data file in a surface layout form (Fig. 6.1) to Algor. This model provides useful basic geometry for constructing the finite element mesh in Algor.

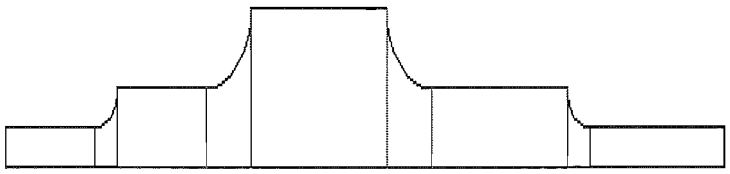


Fig. 6.1 Shaft Geometry File in DXF Format

step 2: mesh the cross section of shaft geometry model (Fig. 6.2).

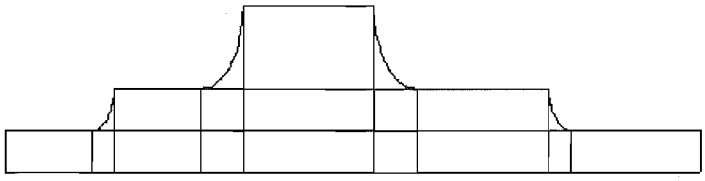


Fig. 6.2 Meshed Shaft Geometry File

step 3: a full mesh is generated by using geometric manipulation tools in Algor to rotate the meshed surface finite element model (Fig. 6.3).

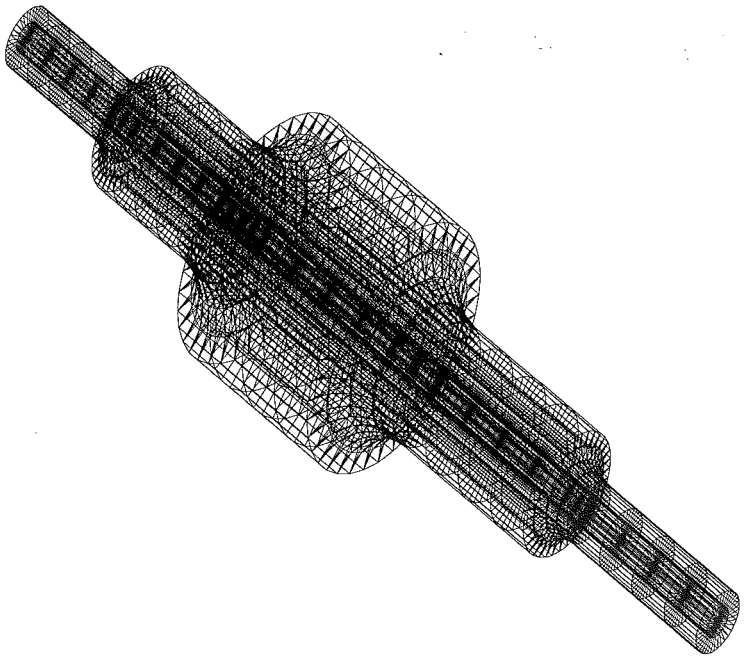


Fig. 6.3 Shaft Finite Element Model

For the shaft keyway section, a mesh is produced by splitting the section into two parts. The whole finite element model is formed by projecting the meshed surface model and connecting the two separate parts (Fig.6.4(a)-(d)).

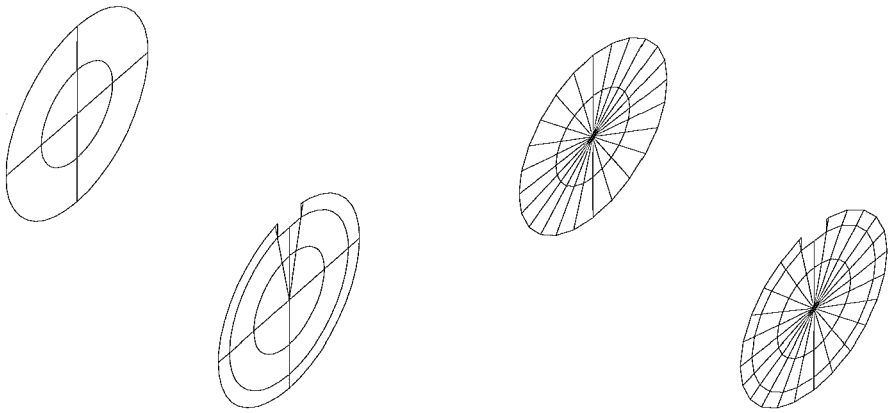


Fig.6.4(a) Cross Sections of Keyway Section Fig.6.4(b) Meshing the Cross Sections

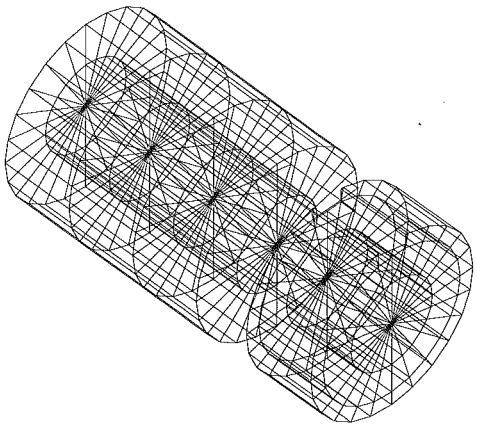


Fig.6.4(c) Projecting the Meshed Cross Sections

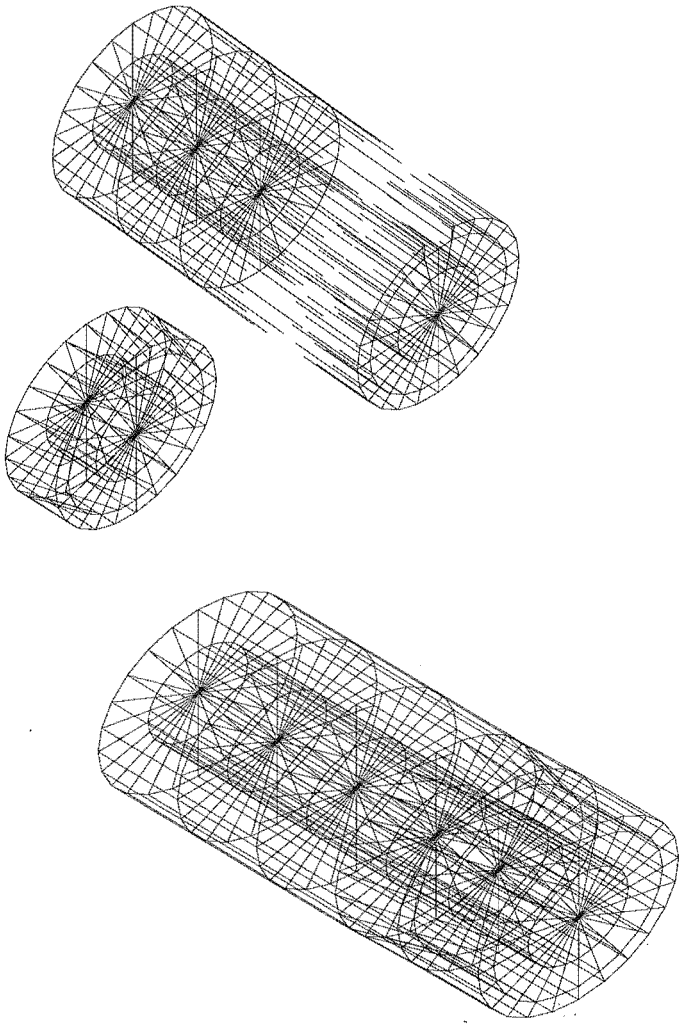


Fig. 6.4 (d) Connecting the Two Components (top) and Finite Element Model for a Shaft with Keyway (bottom)

The interface between MicroStation and MasterCAM is used for passing the required information to the numerical control software for the addition of manufacturing information. This interface is via either the graphic exchange standard (IGES) or DXF format. It is believed that IGES is the principal neutral format currently in use and has a better exchange manner with less information lost during communication. The process is straightforward, a geometry file in MicroStation is converted into an IGES file

which MasterCAM can import through the IGES converter (Fig.6.5). To avoid information loss and obtain a precise geometry file in the MasterCAM for NC program, the suggested transfer method is similar to file converting between MicroStation and Algor. The detailed layout of a shaft can be transferred into MasterCAM in IGES or DXF format. The shaft geometry file will be created by rotating the layout around an axis. This is done via geometry generation functions in MasterCAM. By executing MasterCAM, the path and other machine tool information is formed into machine code which can be read by the numerical control machine tool controller. The information is sent to the numerical control machine for manufacturing the design solution through an RS-232C interface using the "Transmit Function" in MasterCAM. The transmitting baud rate, parity, number of data bits and number of stop bits must be set before transmitting. For the FANUC milling machine in the University of Canterbury's Mechanical Engineering Department, the data above is 4800, even parity, 7 data bits and two stop bits, respectively.

6.2 External Program Communication in MicroStation

Within the MicroStation environment the MDL programs provide the interaction with the designer and a way in which exchange data may be interfaced. They also provide the communication channel between MDL programs and other external programs which are completely separated from MicroStation via MDL external program communication functions. These functions enable MDL programs to start external programs in native machine code and communicate with these external programs through message queues and shared memory (Intergraph, 1992, MDL Course Guide). This study

implements one module for interprocess communication which in turn required a method to interface other software with the MDL environment.

On a personal computer, an MDL program can start other programs that run in real or protected mode. When an MDL program starts a program running in real mode, it starts the real mode program and waits for it to terminate. When an MDL program starts a program to run in protected mode, the external program can return to MicroStation while still remaining loaded. Protected mode simulates multi-tasking (Intergraph, 1992, MDL Course Guide).

In the shaft design process, the shaft feature design is related with the bearing selection which is implemented as a separate module written in Prolog language. It runs in real mode. The MDL side does most of the user interface. The bearing selection box opens along with the shaft bearing seat description input dialog box (Fig.6.6). The designer can push the 'run' button in the bearing selection box to run bearing selection module. The program activates the bearing selection module by calling the mdlExternal_startRealMode Program and waits until the module finishes. At this stage the MicroStation screen is hidden by the bearing selection screen. When the bearing selection module has performed all of its roles, selection data exits, the screen returns to MicroStation automatically.

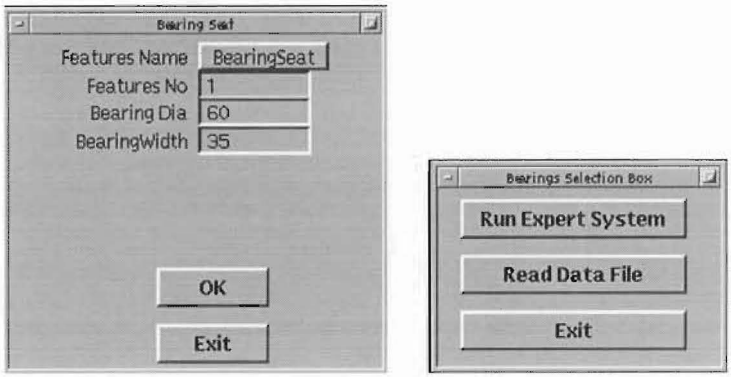


Fig. 6.6 Bearing Seat and Bearing Selection Box

The channel of data transfer between the MDL program and the bearing selection module has been investigated. The simplest method is by means of ASCII text files, which are read in and written out by either program when required (Fig.6.7). The details are described as follows. The MDL program writes the basic bearing data, such as reactions acting on the shaft, shaft diameter, speed of rotation of the shaft and bearing life hours etc. which are collected from the MDL program and the designer, as its output into data file A and the bearing selection module then reads in data file A as an input. Likewise, the bearing selection module writes its output, that is the selected bearing dimensions, to data file B and the MDL program reads the data in data file B as input, this defines the dimensions of one of shaft features - the bearing seat. Fig.6.7 is a simple flow chart to illustrate this process.

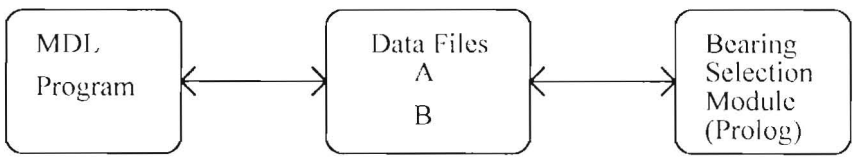


Fig.6.7 Flow Chart of Communication between MDL Program and External Program

Each side when active controls the action of communication eg. reading and writing data via functions within the programs themselves.

The presented communication technique was found to be a simple, clear and efficient way to communicate between MicroStation programs and external programs written in other languages. The main advantage was that the communication interface did not require the linking of two different source codes and the communication of data through external functions as such the technique is language independent.

Chapter 7 Knowledge-based System

Engineers have made extensive use of computers in computer-aided analysis, design and manufacture for more than 20 years. However, the assistance of computers in engineering problem solving has been limited to fast "number crunching" (Chee-Kiong Soh and Ai-Kah Soh, 1990, p7). Developments in artificial intelligence technology have allowed this assistance to extend to problem solving based on the knowledge or expertise of others. Applying artificial intelligence technology to engineering problems is carried out in the form of development of knowledge based systems. With the help of the knowledge-based system, the knowledge and application of the principles of engineering and expertise can be loaded and stored in a computer in the form of facts and rules. The systems then act as design assistants to aid the engineer when making decisions in a design process based on these facts and rules. This study has developed a framework of knowledge-based modules, that assist the designer in selecting bearings in the shaft design process and show the basic concept of how knowledge based systems are applied to engineering design. This chapter addresses the simple application of artificial intelligence to the selection of bearings in the shaft design process.

7.1 Introduction

In recent years, a number of researchers have tried to develop expert systems and software which are useful in the design and selection of

mechanical engineering components. This software provides a knowledge-based aid for the design of mechanical systems. A number of languages and tools are currently available for building the knowledge based systems. Umaretiya and Joshi grouped these tools into three categories (Umareiya and Joshi, 1992, p151) :

1.General purpose programming languages which are used to build current expert system framework, such as LISP and PROLOG.

2. General purpose representation languages which were developed specifically for knowledge engineering, such as SRL,RLL,OPS5 and LOOPS.

3. Domain independent expert system frameworks which provide a system builder with an inference mechanism, knowledge acquisition, and explanation modules to simplify the construction of expert systems, such as EMYCIN,KAS,HERSAYIII, EXPERT, etc.

The selection of the tools and techniques which are used to develop knowledge based systems for engineering design depends on the nature of the engineering problem and intended solution strategy (M.L. Maher, D. Sriram and S.J. Fenves, 1984). An attempt is made in this study to develop a knowledge based system to aid in the task of bearing selection. The system is built as one of the functional modules in the shaft design process, this system is written in Prolog. The main aim of this functional module is to search for techniques to aid mechanical design and to develop an interface between MDL programs and external programs. The following section will present an insight into the bearing selection function module.

7.2 Architecture of the Bearing Selection Module

This module is used for the selection of bearings which are used in a shaft design circuit. The module assists the designer to select the right bearings by using physical and mechanical constraints and formulations.

The system provides the following functions:

1. Bearing Type Selection: Each type of bearing has characteristic properties. The characteristic properties are used to evaluate the suitability of different types of bearing. Eight aspects of the characteristics are considered for this purpose. In order to make the selection as simple as possible, location and misalignment problems have been ignored. Various grades are used for the characteristics criteria, such as excellent, good, fair, poor, etc. The selection of bearing types for certain applications can be made based on the evaluation of the characteristics of the specific bearing type.

2. Bearing Size Selection: This function allows one or a pair of bearings to be selected on the basis of their load carrying capacity and the requirements regarding life and reliability. The user can make a final choice if more then one bearing satisfies the requirements.

3. The bearing selection module displays the result and writes it into a file for later use in MicroStation.

4. The module also provides a channel to examine the catalogue of all bearing types available in the knowledge base.

In the module, the data for the selection is input in a qualitative way, and the computer then compares the requirement with the information it holds on the different bearings to lead to a solution. The main menu and the flow chart for the bearing selection menu are shown in Fig. 7.1 and Fig. 7.2.

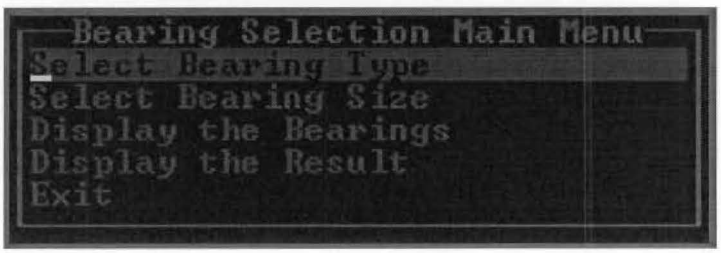


Fig. 7.1 Main Menu for Bearing Selection Module

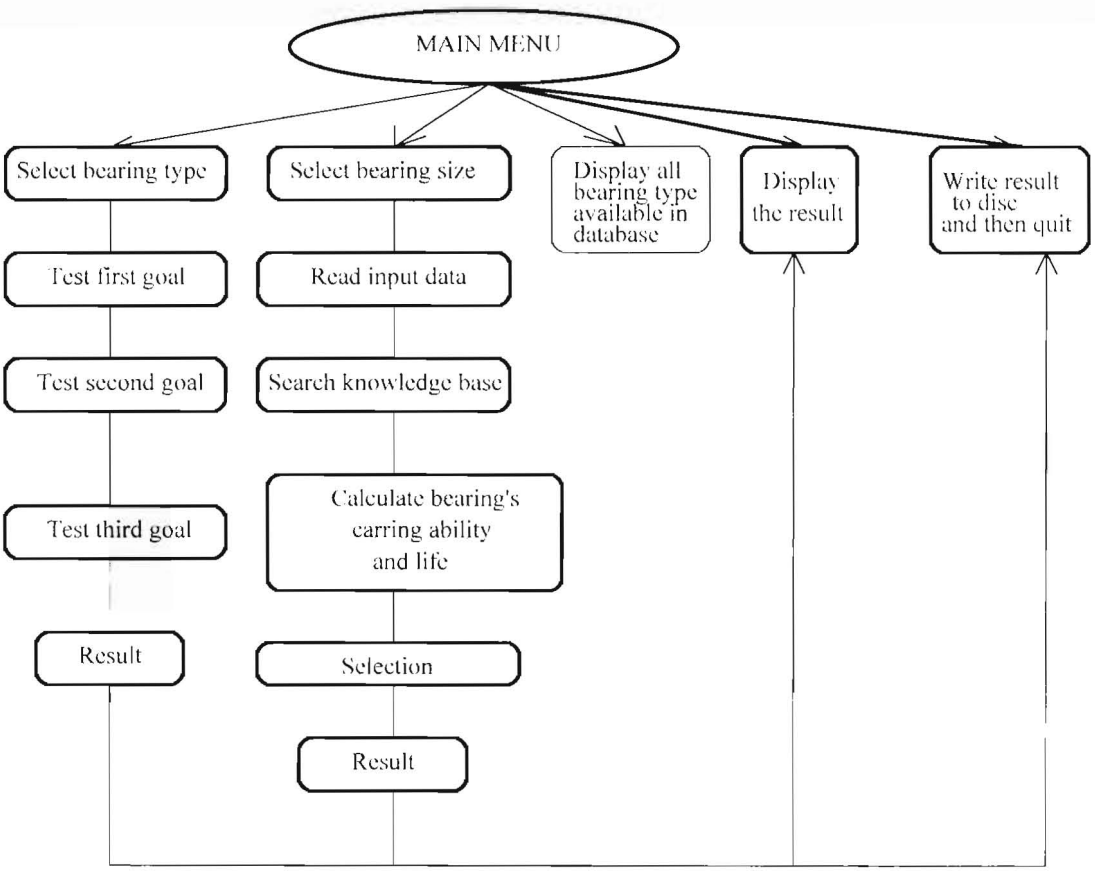


Fig. 7.2 Flow Chart for Bearing Selection Module

Normally, a knowledge based system has three major components. They are: the user interface, knowledge base and an inference engine. This module also employs a dynamic database to store the data coming out during the selection process.

7.2.1 User Interface

The user interface is designed to facilitate communication between the module and the user. It is the means by which the system asks the pertinent questions, to gather the relevant information from the user and communicates the collected information back to the user. All necessary information defining a problem is received through the user interface. The user interface passes this information to the inference engine based on the type and nature of the information (Khin Maung Yin and David Solomon, 1987, p389). The inference engine processes the information, it checks the knowledge base to see any conclusion can be inferred or if more enquires are needed. The conclusion can then be passed back to the user or the user can be prompted for more information through the user interface.

For the bearing type selection, the information is gathered on an attribute basis. That is each characteristic is given an attribute priority according to its importance in the actual application. Then the specification is defined to each characteristic by using grades as mentioned earlier. An example of user interface is shown in Fig. 7.3.

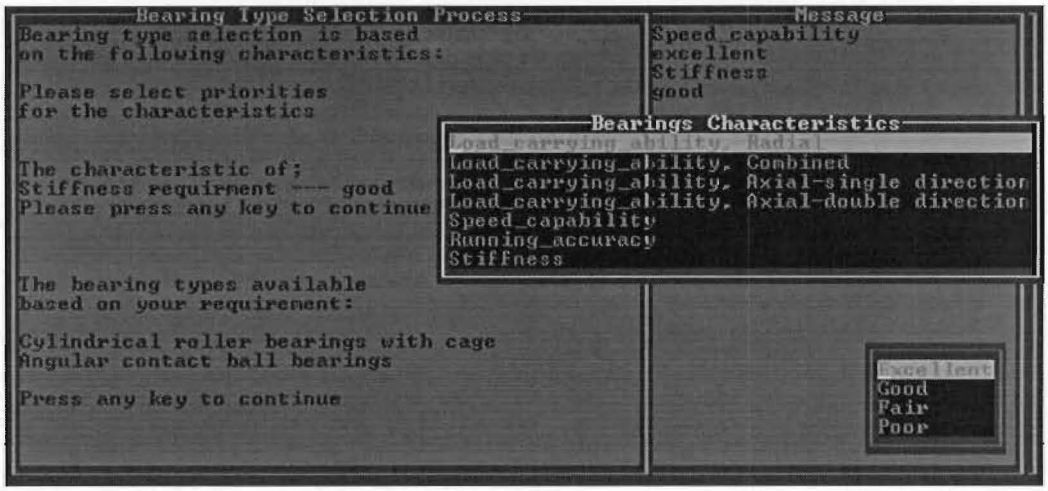


Fig. 7.3 An Example of User Interface for Bearing Type Selection

For the bearing size selection, the interface collects input specifications which typically include the diameter of the shaft, the load, the speed of rotation of the shat, the bearing basic rating life, and so on. The data is read from the file and displayed through the interface on the screen. The user may change the data if necessary (Fig. 7.4).

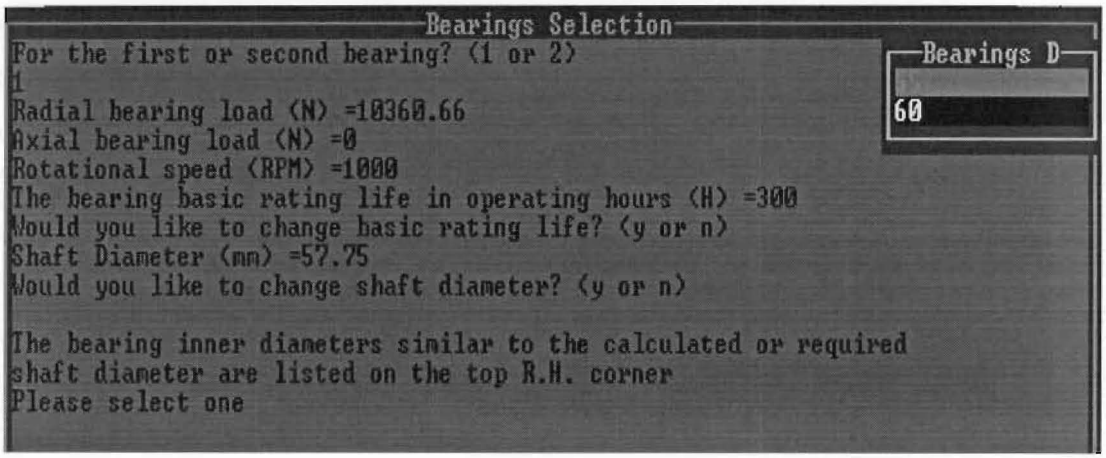


Fig. 7.4 User Interface for Bearing Size Selection

7.2.2 Knowledge Base

A domain-specific knowledge base has been developed to aid the inference process. It is the central part of the bearing selection module. It contains rules describing the relations and physical properties of the bearings. The knowledge needs to be represented in a form that can be used for efficient storage, retrieval and reasoning. Many types of knowledge structure are employed to form a knowledge base, such as "facts, frames, procedures, rules, and semantic networks" (Umaretiya and Joshi, 1992, p154). For the bearing selection module, frames are more suited than other representations. An object can be described by a collection of faces called a frame. A frame is a list of attributes under a common label, so that the information about an entity can be grouped into an unfirm structure. The different properties of an entity are each represented by a slot and a value. Each slot represents an attribute of an object. The structure for frames is:

Frame Name
(slot 1) (value)
(slot 2) (value)
.....

In the case of bearing type selection, the frame in the module is the bearing type frame. That is the most fundamental properties of each bearing type are represented in frames, so frames hold all of the different bearings characteristics and their values. Each frame corresponds to one characteristic of each type. The basic structure of the frame is defined as follows:

Bearing(type name,"characteristics", characteristic name, value)

An example of a frame of the above description is:

bearing("Deep groove ball bearing","characteristics","Load-carrying-ability","fair")

In the case of bearing size selection, the frame contains all the basic information for one particular bearing. The structure of the frame is:

frame(type no,designation, internal inner race dia., external outer race dia., width, basic load rating-dynamic, basic load rating-static, external inner race dia.)

eg. frame(1,"16011",55,90,11,15000,12200,67)

With all of the bearings grouped according to their type, they may be divided into smaller groups based on their internal inner race dia. The searching process may be more efficient if the bearing data is analysed on a group basis and then using size criteria.

7.2.3 Searching Strategy - Inference Engine

Based on the information received from the user, the inference engine reasons over the knowledge base to arrive at conclusions. It is a set of reasoning processes and employs a kind of simple problem solving technique. The inference engine consists of operating rules and principles. It executes the these rules and determines when an acceptable solution has been found (Khin Maung Yin and David Solomon, 1987, p389).

The searching process for bearing type selection is invoked by requesting a goal to be satisfied. Through the user interface, the first goal is formed based on the characteristic which is given the top priority in the

selection process, and the specification of the characteristic. The program then interrogates the knowledge base to determine whether a particular bearing type can achieve this goal. The inference process is performed by means of searching and pattern-matching. The first objective of the process is to match the goal with frames in the knowledge base. The inference engine involves scanning down the list of frames in the knowledge base until the whole knowledge base is scanned. After the list of possible bearings is identified, the program selects the bearings which best match the specification. If one or more matching frames are found, these frames which satisfy the first goal will form a shortlist. The program will then ask the user to select another characteristic, with specifications, to form the second goal. The inference process will continue to match the second goal with the frames in the shortlist. The process is repeated until the final solution is found. If no match is found, the interface asks user to weaken or chose another specification and repeat the process. The structure chart for the bearing type selection is shown in Fig. 7.5.

The bearing type selection is the first stage in the bearing selection module, subsequent stages then select particular bearings using detailed quantitative specifications, that is the bearing size selection. The flow chart for bearing size selection is shown in Fig. 7.6.

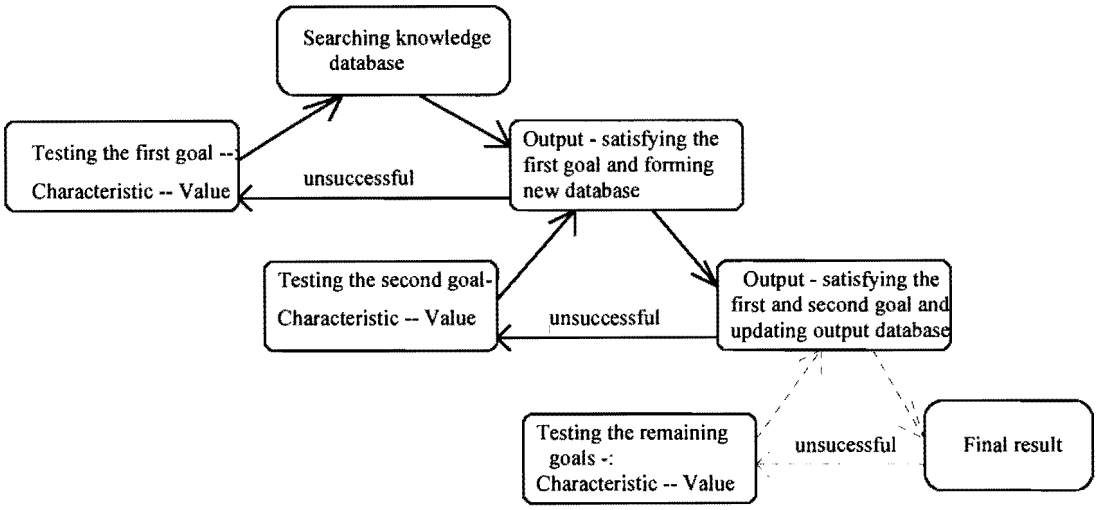


Fig. 7.5 Searching Process for Bearing Type Selection

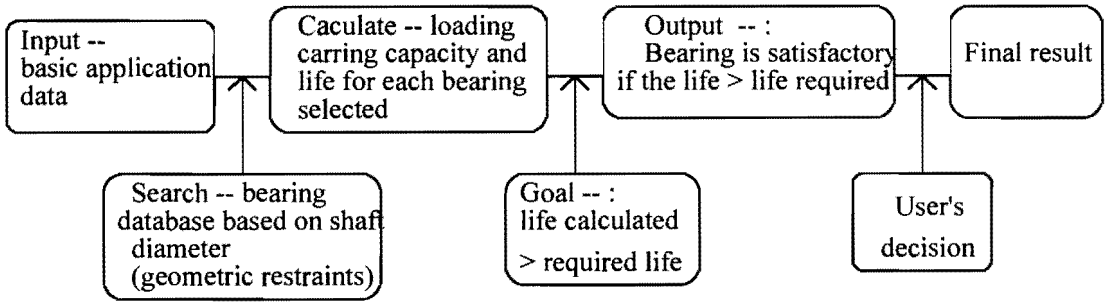


Fig. 7.6 Flow Chart for Bearing Size Selection

As mentioned in 7.2.1, certain specifications are required to make a bearing selection. This data including radial and axial bearing loads, shaft rational speed, shaft diameter and the required bearing basic rating life are gathered as input through the shaft design user interface in the MDL program described in Chapter 3. This data is stored in a file on disc for the bearing selection module to read. The module selects one bearing each time. Two bearings sets are needed to support a shaft. A question is asked to decide if the data for the first or second bearing is needed. The data is displayed through the bearing selection interface. Questions are asked to confirm the data, and values can be re-assigned if necessary. The required operating hours and shaft

diameter form the constrains for the selection, so that the goal of the selection is to select a bearing with a sufficient basic dynamic load rating to meet the life requirement and to satisfy the constraints related to the diameter of the shaft, loads, and so on. The selection procedure enables a user first to narrow down the list of the bearings available by ranking the alternatives by shaft diameter so as to create a shortlist of acceptable bearings. For these bearings, the equivalent dynamic load acting on the bearing must first be calculated from the radial and axial bearing loads and then the basic rating life of each bearing is calculated. The formulas for the above are described in Appendix F. The selection process then compares the life calculated with the life required for the bearings which satisfy the constraint related to shaft diameter. If the life calculated is more than the required life, the bearing is satisfactory. The output at this stage may contain more than one bearing. In this case, the choice of one bearing is required from the user from the list of bearings which are suitable for the requirement. If there is no result eg. no acceptable solutions the constraints need to be re-defined, eg. changing the shaft diameter, then the selection process may be repeated.

7.3 Database and File Processing

The knowledge base is actually a database. It represents a collection of facts in a particular area of knowledge and the relationships between these facts. There are two types of database in the module: static and dynamic database. The static database contains all 'fixed' knowledge. When executing the module the knowledge base does not change. The module uses the dynamic database to handle the matching frames coming from the knowledge

base and the input data. The database can be manipulated as the module is executed. This database is always updated when new matching frames are found or the input data is re-assigned. The final outcome is stored in this database.

The bearing selection module is part of the shaft design assistant system. It preforms one particular activity. It should be able to communicate and transfer data with other separate modules in the system. This function is achieved through file processing. By using facilities for file processing provided by Prolog, the data can be read from and written to the common disc file shared by separate modules. This is one of the simple, convenient and effective ways for communication between modules. Through file processing, the bearing selection module reads expected loading and operating data from the disc file as its input and writes a file of selected bearing data, eg. internal inner race dia., width, external outer race dia., etc. to the disc.

7.4 Discussion

As an artificial intelligence approach to an engineering problem, the system should have some important capabilities, for example, the system should provide the explanations for the adopted solution or procedure inferred from the knowledge base. It can propose some advice to the engineer at any stage and assist the engineer to reach some sort of compromise when there is no solution reasoned over the knowledge base. The focus of the study has been on the interface and communication between the user and modules. As an artificial intelligence design assistant, the module needs more development

as mentioned above. The module uses a very limited database to test the basic principles of the bearing selection process. If the module is to be of use, it needs an increased database with more selection rules related to factors upon which the choice of bearing can depend, so more effort is required to convert the demonstration program into a realistic system.

Chapter 8 Example

This chapter provides an example of a session with the shaft design assistant the program. It is used to demonstrate the main features of th program. The general steps to run the program are listed as follows.

1. prepare the shaft load and bearing basic data,

2. load the shaft design assistant program in the MicroStation environment,

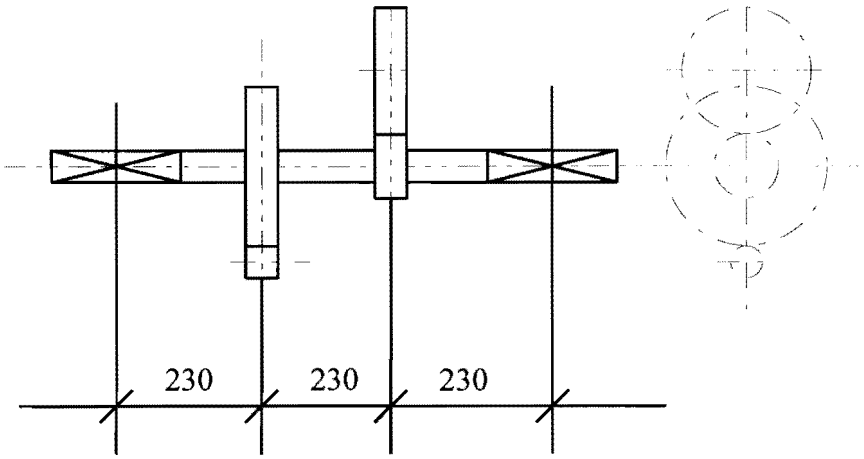
3. enter the input data into the Input Information Box,

4. interact the functions embedded in the program according to the stage in the shaft design process.

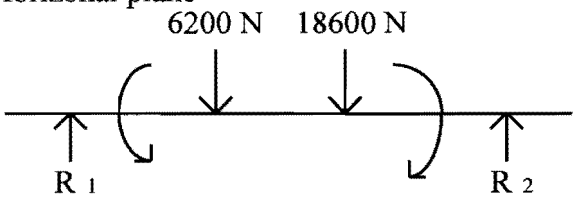
8.1 Example Specification

The specification for the example is described in the following drawing:

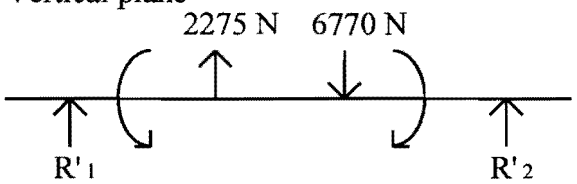
Double - reduction train of gears



Horizonal plane



Vertical plane



T=1860,000 Nmm

Material AISI 4340

Bearing data: rotating speed rpm= 1000
bearing basic rating life in operating hours L10=200h

8.2 Running the Shaft Design Assistant Program

1. The load and geometry data is input as shown in Fig.8.1.

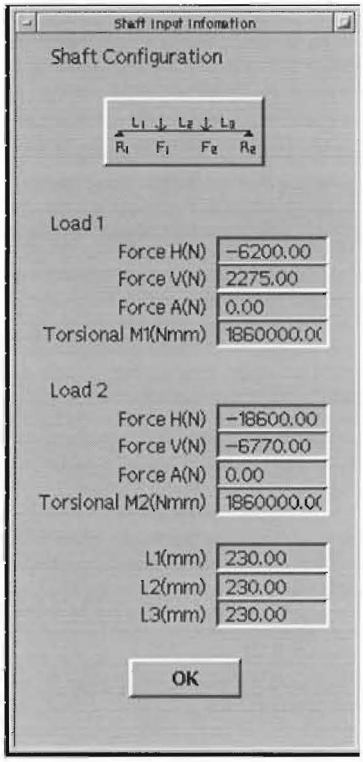


Fig.8.1 Shaft load and geometry data

2. The material and safety factor data are chosen by the user. The bearing's reactions are calculated by the bearing reaction calculation function. The data is then displayed in the Shaft Information Box as shown in Fig.8.2. The basic life of the bearing and shaft rotating speed are input into the this box.

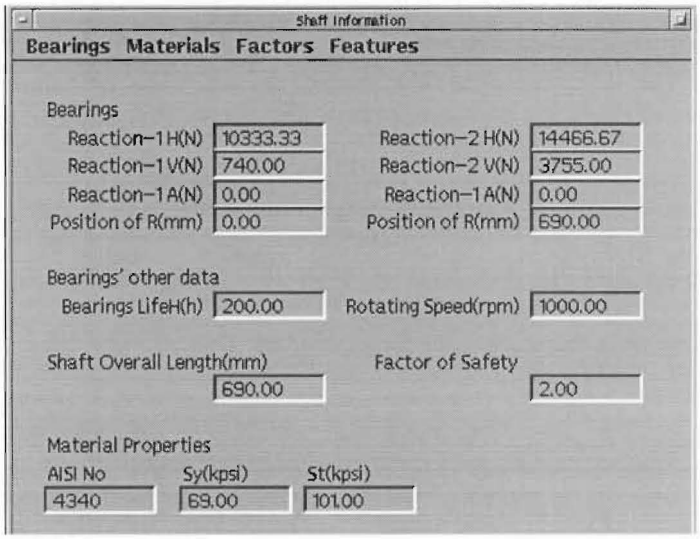


Fig.8.2 Bearing reactions, material and factor of safety data

3. Fig.8.3 shows shaft layout executed by the Draw a Sketch Module.

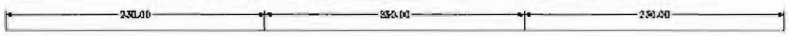


Fig.8.3 Shaft layout with dimensions

4. Shaft basic diameter is calculated based on failure under peak load and a drawing with the basic diameter is displayed (Fig. 8.4).

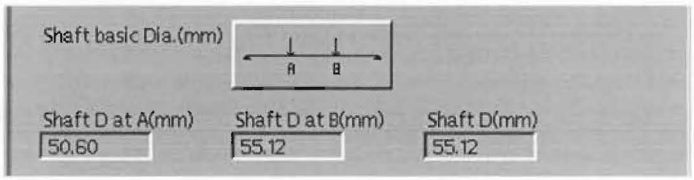


Fig.8.4 (a) Shaft basic diameter

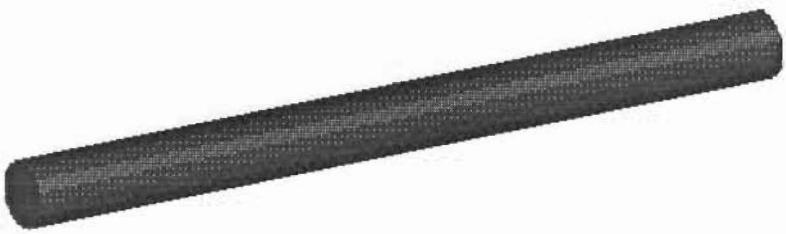


Fig.8.4 (b) Shaft with basic diameter

5. Open Shaft Feature Selection Box to start shaft detail design.

6. Start bearing selection. The Bearing Selection Module is initiated by the External Program Communication Module (Fig.8.5).

Fig.8.5 Bearing Selection Box for Communication of
Bearing Selection Module in Prolog

7. The bearing data is read by the program through the disc file (Fig.8.6).

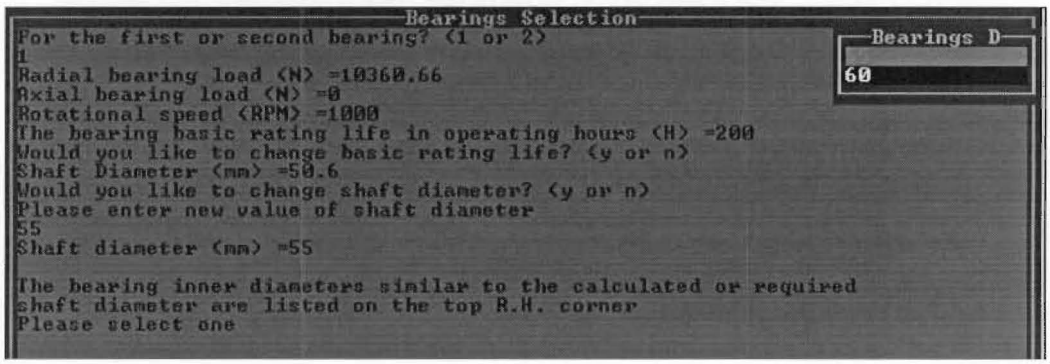


Fig.8.6 Input of bearing basic data

8. The available bearings which satisfy the requirements are shown in Fig.8.7 (a). Finally, one bearing is chosen and the data is written to the disc file (Fig.8.7 (b)).

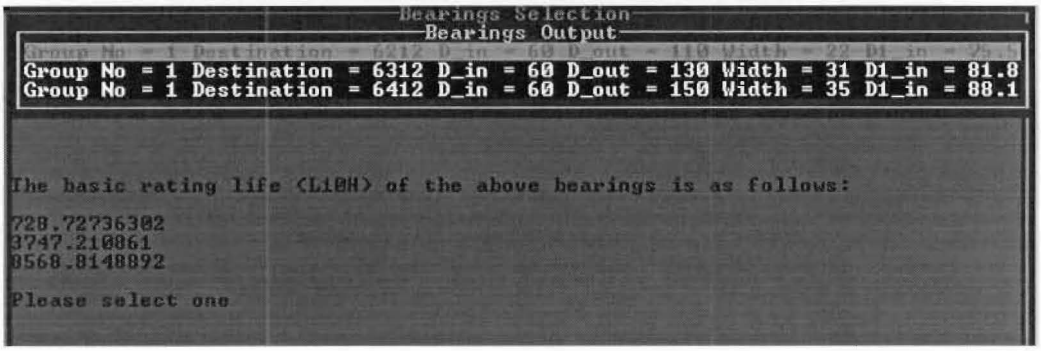


Fig.8.7 (a) The available bearings

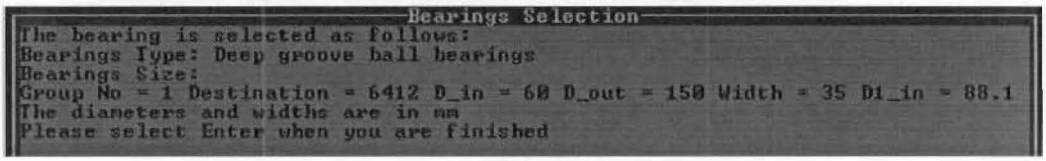


Fig.8.7 (b) Final selected bearing data

9. The bearing data is read through the disc file to the bearing seat box (Fig8.8).

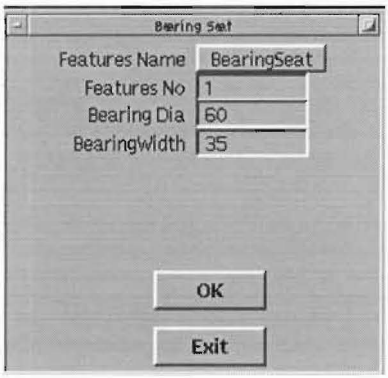


Fig.8.8 Bearing seat data

8. Shaft feature detail design data and the finished shaft drawing is shown in Fig.8.9.

Fig.8.9 (a) Design of shaft feature

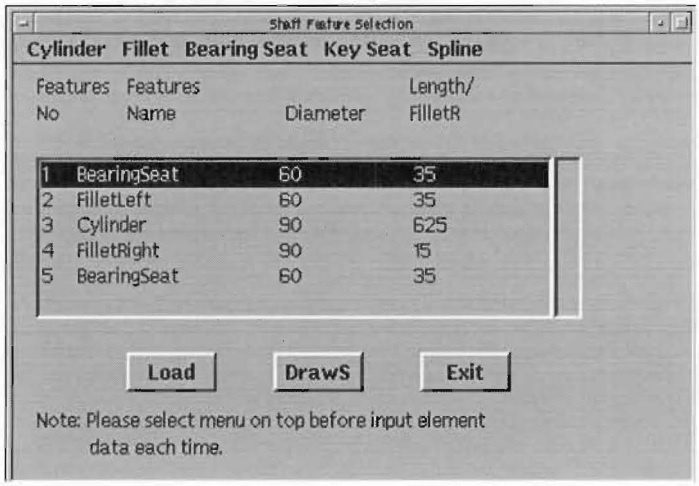


Fig.8.9 (b) The whole shaft data

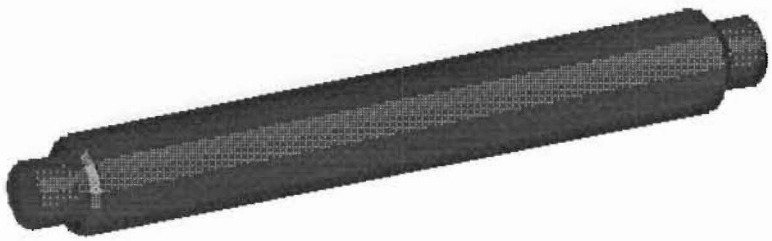


Fig.8.9 (c) Shaft drawing

9. The DXF geometry file is transferred to Algor and MasterCAM (Fig.8.10).

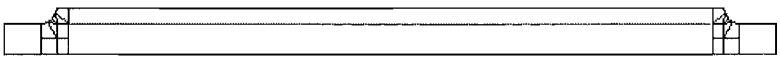


Fig.8.10 (a) Geometry file in DXF format to Algor

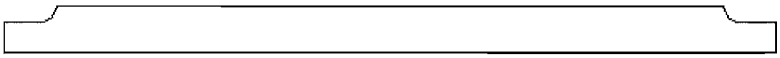


Fig. 8.10 (b) Geometry file in DXF format to MasterCAM

Chapter 9 Conclusion and Further Development

This thesis has presented some work on the interface between users and software based on the MicroStation environment. The design of shafts and their associated components was the main system for this investigation. A shaft design assistant program, written in MDL and Prolog, is reported in this thesis. However, this project aims to demonstrate interface methods and techniques rather than provide a truly viable shaft design assistant system. More development is required both in terms of interface methods and the shaft design assistant program.

9.1 Conclusion

Channels of interface between software in a local environment, especially among MicroStation, Algor, and MasterCAM and also between MicroStation and programs written in other computer languages are investigated in this study. The communication between the software packages is mainly through common data files in a certain format. DXF and IGES formats are popular and acceptable among the CAD/CAM packages which share the product geometry files. Although complete geometric information of the product is available for the exchange between the CAD/CAM software packages, the simplified or abstracted form of the relevant geometry is usually considered for the transition in order to get an idealised model suitable for

finite analysis in Algor and manufacturing in MasterCAM. The transferred geometry is then reconstructed using geometry generation functions in Algor and MasterCAM. The knowledge-based approach has been applied to engineering component selection. The program based on this approach is given as an example of communication between MDL applications and programs written in the other languages. The MDL application activates the program using Prolog in the MicroStation environment. The input and output of the program are channelled to disc ASCII files. Sharing the common ASCII file between the MDL applications and programs in other languages like Prolog is believed to be the best way of communication.

This study also demonstrates that the application of MDL is a feasible approach for users to create their own system. MDL provides the powerful tools which are viable aids to application development. Through MDL application, all MicroStation functions can be called to interface with the user, create and manipulate graphic elements and synchronise with MicroStation and other applications either in or out of the MicroStation environment. The application can be used just like embodied functions in MicroStation, hence users may take full advantage of MicroStation's "CAD engine" for their own use. The use of interfaces makes the application versatile. MDL provides the facilities required to develop a graphical user interface which implements some of the common features of a user interface such as menus, icons, message areas etc. Through the interface the user can control and communicate with the application and launch the user's own functions, and the user's input can be processed. Several standard dialog items in MicroStation have been examined to contract the graphic interfaces. Several MDL build-up functions have also been used to assist the shaft design process, such as

dimension, element creation, element description, external program communication, stringlist, user interface functions and so on.

9.2 Limitations of the Study

There were some limitations encountered while programming the system.

1. MicroStation Version 4 does not support the tool for trimming normal surfaces, only the trimming B-Spline surface function is provided. Hence the drawing functions in MDL are confined by this limitation. The keyway part of the shaft can not be drawn properly. However, MicroStation Version 5 has enhanced ability in drawing. Fig.9.1 shows the keyway created by using trimmed surface functions in MicroStation Version 5.

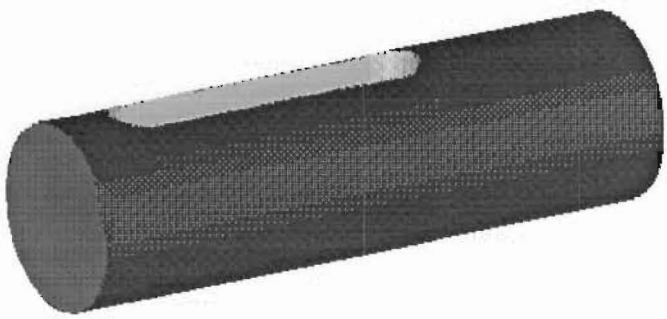


Fig. 9.1 A keyway on the shaft

2. As mentioned earlier, the purpose of this thesis focuses mainly on investigating the interface software as well as the MDL functions in which the interface is achieved. Hence, the system is developed on the principle points of shaft design. The system only contains the basic facilities necessary to aid

the design process. Other than the limited points discussed in 5.4 and 7.4 respectively, there are some other limitations:

a. To simplify the shaft design, no more than two elements other than bearings on the shaft are considered.

b. When deciding on an improved approach to the shaft design, fatigue analysis must be included within the shaft size determination calculations. There is a program related to shaft fatigue analysis written in BASIC (B.M. Foster,1990, D.J. Kennedy,1991). This program has not been involved in this program.

c. There are some standard elements which are not included in the program due to the limitation of drawing facilities in MicroStation Version 4, such as splines, woodruff keys etc.

3. To form an integrated shaft design assistant, it is better to encompass all of the different modules which perform different functions in the design process, such as, drawing, analysis, and manufacturing within a "window operating system". The modules interface with each other through the windows environment. This is restricted by the size of memory the software must use when active.

9.3 Further Development

As mentioned above, the program is treated as an experimental and investigating tool rather than a useful shaft design assistant, further development of the system is definitely required if it is to be implemented as a shaft design assistant and as a application of MDL. If the same environment is still adopted, the following points are the most promising areas for further development other than the areas mentioned in 9.2.

1. Using an expert system environment to control the shaft design system would take advantage of knowledge and application of the principles of engineering and expertise. The knowledge can cover computer aided design and computer aided manufacturing. This will enhance the systems ability as a design assistant.

2. Database control and management is another area needing more development. The channel to Oracle and dBase in the MicroStation environment makes it possible for the system to handle and access the large database used in the design process.

REFERENCES

Abdou, G.H., 1992, Integrated Approach to Knowledge-based Process-plan Generation, *Knowledge-based Systems*, Vol.5, No.4, pp269-276.

Allada, V. and Anand, S., 1992, Manufacturing Applications of Octrees, *Computers & Industrial Engineering*, Vol.23, Nov. pp37-40.

Bertini. L, 1993, A Prototype Expert Systems for Embodiment Design of Mechanisms and Articulated Systems, *Artificial Intelligence in Engineering*, Vol.8, No.1, pp57-65.

Bradbeer, P.E., 1992, Design and Development of an IGES Editor Using Ada, *Ada in Transition 1992 Ada UK, International Conference Proceedings*, London, UK, pp120-30.

Bzymek, Z.M. and Joo Jung, 1990, an Intelligent System for Design of Electric Motors, *Proceedings of the Second IASTED International Symposium Expert Systems and Neural Network*, HI, USA, pp158-161.

Capelo, A.C., Ironi, L. and Tentoni, S., 1992, a Tool for the Study of Materials Combining Qualitative and Quantitative Models, *Twelfth International Conference. Artificial Intelligence, Expert Systems, Natural Language*, Nanterre, France, pp689-95, Vol.2.

Capelo, A.C. Ironi, L. and Tentoni, S., 1993, A Model-based System for the Classification and Analysis of Materials, *Intelligent Systems Engineering*, Vol.2, No.3, pp145-58.

Case K. and Gao J., Feature Technology; an Overview, 1993, *International Journal Computer Intergrated Manufacturing*, Vol.6, No.1, pp2-12.

Cavendish, J.C., Frey, W.H. and Marin, S.P., 1991, Feature-based Design and Finite Element Mesh Generation for Functional Surfaces, *Advances in Engineering Software and Workstations*, Vol.13, No.5-6, pp226-37.

Cha, J., Rao, M. and Zhao, J., 1993, Integrated Intelligent Environment for Gear Manufacturing System, *Applied Artificial Intelligence*, Vol.7, No.2, pp177-93.

Chakrabarti, A., Bligh, T.P. and Holden, T., 1992, Towards a Decision-support Framework for the Embodiment Phase of Mechanical Design, *Artificial Intelligence in Engineering*, Vol.7, No.1, pp21-36.

Chen, C.L.P. and Wichman, C.A., 1992, A Systematic Approach for Design and Planning of Mechanical Assemblies, *[AI EDAM] Artificial Intelligence for Engineering Design, Aanalysis and Manufacturing*, Vol.7, No.1, pp19-36.

Colebourne, A., Sawyer, P. and Sommerville, I., 1993, MOG User Interface Builder: a Mechanism for Integrating Application and User Interface, *Interacting with Computers*, Vol.5, No.3, pp315-31.

Denzel, H. and Vosniakos, G.-C., 1993, a Feature-based Design System and its Potential to Unify CAD and CAM, *IFIP Transactions B [Applications in Technology]*, Vol.B-10, pp131-44.

Dilger, W., 1991, Computer Aided Eliciting of Design Knowledge on the Basis of Relational Representations of Geometrical Objects, *Proceedings Fourth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-91)*, Kauai, HI, USA, pp720-9.

Dong,Z. and Hu, W.,1991, Candidate Machining Sequence Generation for Optimal Process Planning Using a Knowledge-based System, *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Victoria, BC, Canada, Vol.2, pp569-72.

Douglas, S., Doerry, E. and Novick, D., 1990, QUICK: a User-interfae Design Kit for Non-programmers, *Proceedings of the ACM SIGGRAPH Symposium*, Snowbird, UT,USA, pp47-56.

Duan, Weiying, Zhou, Ji and Lai, Kewei, 1991, Feature Oriented Modeling Tool: New Foundation of Feature-based Design, *Second International Conference on Computer-Aided Design and Computer Graphics*, Hangzhou, China, pp252-7.

Duan, W., Zjou, J. and Lai, K., 1993, FSMT: a Feature Solid-modelling Tool for Feature-based Design and Manufacture, *Computer Aided Design*, Vol.25, No.1, pp29-38.

Duhovnik, L. and Zavbi, R., 1992, Expert Systems on Conceptual Phase of Mechanical Engineering Design, *Artificial Intelligence in Engineering*, Vol.7, No.1, pp37-46.

Fagan, Michael J., 1987, Expert Systems Applied to Mechanical Engineering Design, *Computer-aided Design*, Vil.19, No.7, pp.

Flurscheim, Charles, 1977, Engineering Design Interfaces - a Management Philosophy, Design Council.

Haasos, S., A Composite Design System Based on Knowledge, 1993, *ZWF Zeitschrift fur Wirtschaftliche Fertigung und Automatisierung*, Vol.88, No.55, pp218-21.

Held, H.-J., Jager, K.-W, Kratz, N. and Schneider, M., 1991, Knowledge-based Engineering Assistance, *Artificial Intelligence in Design '91*, Edinburgh, UK, pp883-95.

Helpenstein, H.J., 1993, Applications of Neutral CAD Interfaces to FEM Systems, *VDI-Z*, No.Spec., May, 1993, pp22-7.

Helpenstein, H.J. and Heinrichs, H., 1992, STEP Processors for CAD Data Exchange, *CARs and FOF. 8th International Conference on CAD/CAM, Robotics and Factories of the Future*, Metz, France, pp499-507.

Hemmelgarn, D. and Hodges, J., 1991, IGES to STEP Migration, *AUTOFACT '91. Conference Proceedings*, Chicago, IL, USA, pp22-11-17.

Hoffman, C.M. and Juan, R., Erep-an Editable, 1992, High-level Representation for Geometric Design and Analysis, *IFIP Transactions B [Applications in Technology],* Vol.B-9, pp129-64.

Horvath, I., Kulcsar, P. and Horvath, L., 1993, Concurrent Engineering with the Prodes System, *IFIP Transactions B [Applications in Technology]*, Vol.B-11, pp105-16.

Hyowon Suh, Ahluwalia, S., and Miller, J.E., 1991, Feature Generation in Concurrent Engineering Environment, *Proceedings of Symposium on Solid Modeling Foundations and CAD/CAM Applications*, Austin, TX, USA, pp493-502.

Intergraph, 1991, MDL Manual.

Intergrph, 1992, MDL Course Guide.

Jones, Peter F., 1992, CAD/CAM Features, Applications and Management, The Macmillan Press Ltd.

Kalta, M. and Davies, B.J., 1993, Guidelines of Building 2D CAD Models of Turned Components in CAD-CAPP Integration, *International Journal of Advanced Manufacturing Technology*, Vol.8, No.5, pp285-96.

Kalta, M. and Davies, 1993, B.J., Converting 80-character ASCII IGES Sequential Files into More Conveniently Accessible Direct-access Files, International Journal of Advanced Manufacturing Technology, Vol.8, No.3, pp129-44.

Kar Tshou Leong, Siang Kok Sim, Yiu Wing Chan, 1991, BESMED: A Blackboard Knowledge-based Approach to Integrate Mechanical Design, *Engineering Applications of Artificial Intelligence*, Vol.4, No.3, pp205-20.

Kim, J.-Y., Mittal, R.O., O'Grady, R.M.P. and Young, R.E., 1992, Process Selection for Concurrent Engineering in the Domain of Rotational Parts, *Journal of Design and Manufacturing*, Vol.2, No.4, pp199-209.

Kim, Steven H., 1991, Knowledge Systems Through PROLOG, Oxford University Press.

Kirk, J.W., Thu Hua Liu and Fischer, G.W., 1992, PDES/STEP-based Information Model for CAE and CAM Integration, *International Journal for Systems Automation: Research & Applications*, Vol.2, No.4, pp375-94.

Kiruchi, Y., Kishinami, T. and Saito, K., 1992, Exchange of CAD Data Using a Three Schema Architecture, *Journal of the Japan Society of Precision Engineering*, Vol.58, No.1, pp145-150.

Klueh, D.W. and Cashman, J.E., 1991, Opening Mechanical Design Automation via Industry Standards, *AUTOFACT '91. Conference Proceedings*, Chicago, IL, USA., pp22-19-30.

Kojima, T., Nakamura, I., Kugai, Y. and Kimura, F., 1993, A Study on the Formulation of CAD Data Exchange System Using STEP, *Journal of the Japan Society of Precision Engineering*, Vol.59, No.2, pp239-244.

Kumar, B., Anand, D.K., Anjanappa, M. and Kirk, J.A., 1993, Feature Extraction and Validation within a Flexible Manufacturing Protocol, *Knowledge-based Systems*, Vol.6, No.3, pp130-40.

Laakko, T. and Mantyla, M., 1993, Feature Modelling by Incremental Feature Recognition, *Computer Aided Design*, Vol.25, No.8, pp479-92.

Lai, S.H.-Y, 1993, KBDA - a Knowledge Based Design System for Assembly, *Computers & Industrial Engineering*, Vol.25, Sept., pp585-8.

Lakamde, M., 1991, Automotive CAD Data Exchange Issues, *AUTOFACT '91 Conference Proceedings*, Chicago, IL, USA, pp26-1-5.

Lenau, T. and Mu, L., 1993, Features in integrated Modelling of Porducts and their Production, *International Journal of Computer Integrated Manufacturing*, Vol.6, No.1-2, pp65-73.

Liu, Thu-hua and Fishcher, G.W., 1993, Development Feature-based Manufacturing Applications Using PDES/STEP, *Concurrent Engineering: Research and Application*, Vol.1, No.1, pp39-50.

Madural, S.S. and Li, Lin, 1992, Rule-based Automatic Part Feature Extraction and Recongnition from CAD Data, *Computers & Industrial Engineering,* Vol.22, No.1, pp49-62.

Maher, M.L., Sriram, D. and Fenves, S.J., 1984, Tools and Techniques for Knowledge-based Expert System for Engineering Design, *Advanced Engineering Software*, Vol.6, No.4, pp178-88.

Mihara, K., Hirose, A. and Harada, Y.,1992, A Method to Integrate CAD and CAM for the Product Prototype Development, *IFIP Transactions B [Applications in Technology]*, Vol.B-3, pp283-98.

Molloy, E., Yang, H. and Browne, J., 1993, Feature-based Modelling in Design for Assembly, *International Journal of Computer Integrated Manufacturing*, Vol.6, No.1-2, pp119-25.

Motz, D.S. and Haghighi, K., 1991, A Knowledge-based Design Model for Mechanical Components, *Engineering Applications of Artificial Intellegence*, Vol.4, No.5, pp351-58.

Neelamkavil, Francis, 1989, Methods for the Development of Man-machine Interfaces with Application to CIM, *Computer-Aided Engineering Journal*, Vol.6, No.2, pp59-64 .

Negele, A. and Rathke, C., 1991, KNOEX+: An Interactive Design Expert, Human Aspects in Computing Design and Use of Interactive Systems and

Work with Terminals, *Proceedings of the Fourth International Conference on Human-computer Interaction*, Stuttgrat, Germany, pp844-48.

Oh, V. Sommerville, I. Taleb-Bendish, A. and French, M., 1992, SIMAD: a System for Improving Mechanical Assembly Design, *Proceedings of the First Singaport International Conference on Intelligent Systems (SPICIS '92). Intelligent Systems 2000*, Singapore, pp257-62.

Pausch. R., Conway, M. and Deline, R., 1992, Lessons Learned from SUIT the Somple User Interface Toolkit, *ACM Transactions on Information Systems*, Vol.10, No.4, pp320-44.

Pittman, J.H. and Kitrick, C.J., 1990, VUIMS: a Visual User Interface Management System, *Proceedings of the ACM SIGGRAPH, Symposium, Snowbird*, UT, USA, pp36-46.

Qiao, L.H., Zhang, C., Liu, T.H., Wang, H.D.B. and Fischer, G.W., 1993, A PDES/STEP-based Product Data Preparation Procedure for Computer-aided Process Planning, *Computer in Industry*, Vol.21, No.1, pp11-22.

Rao Nalluri, S.R.P. and Gurumoorthy, B., Knowledge Based Gluing Operators for Feature Based Modelling, 1992, *CARs and FOF. 8th International Conference on CAD/CAM, Robotics and Factories of the Future*, Metz, France, Vol.1, pp147-61.

Rosen, D.W. and Dixon, J.R., 1992, Languages for Feature-based Design and Manufacturability Evaluation, *International Journal of Systems Automation: Research & Applications*, Vol.2, No.4, pp353-73.

Rosen, D.W. and Peters, T.J., 1992, Topological Properties that Model Feature-based Representation Conversions within Concurrent Engineering, *Research In Engineering Design*, Vol.4, No.3, pp147-58.

Ruttkay, Zsofia, 1987, Multi-media Presentation in CAD Systems, *Intelligent CAD Systems II, Implement Issues*, Springer -Verlay.

Sakthivel, T.S. and Kalyanaraman, V., 1993, A KBES for Intergrated Engineering, *Engineering with Computers*, Vol.9, pp1-16.

Salomons, O.W., Kappert, J.H., van Slooten, F., van Houten, F.J.A.M. and Kals, H.J.J., 1993, Computer Support in the (Re)design of Mechanical Products. A New Approach in Feature Based Design, Focusing on the Link

with CAPP, *IFIP Transactions B [Applications in Technology]*, Vol. B-11, pp91-103.

Salomons O.W., van Houten F.J.A.M. and Kals, H.J.J., 1993, Review of Research in Feauture-Based Design, *Journal of Manufacturing Systems*, Vol.12, No.2, pp113-32.

Sapidis N. and Perucehice R., 1992, Solid/Solid Classification Operations for Recursive Spatial Decomposion and Domain Triangulation of Solid Models, *Computer Aided Design*, Vol.24, No.10, pp517-13.

Sastry, L., 1992, User Interface Mangagement System for Engineering Application, *Computer Graphics Forum*, Vol.11, No.2, pp113-129.

Schaal S. and Ehrlenspiel K., 1993, Design Concurrent Calculation: A CAD and Data - integrated Approach, *Journal of Engineering Design*, Vol.4, No.2, pp75-88.

Shah, Jami J. and Rogers, Mary T., 1988, Functional Requirements and Conceptual Design of the Feature-based Modelling System, *Computer-Aided Engineering Journal,*Feb. pp9-15

Shigley, Joseph Edward, 1983, Mechanical Engineering Design, 4th ed. McGraw-Hill.

Shneiderman, Ben, 1992, Designing the User Interface: Strategies for Effective Human - Computer Omteractopm. 2nd, Addison - Wesley.

Singh, G. Kok, C.H. and Ngan, T.Y., 1990, Druid: a System for Demonstrational Rapid User Interface Development, *Proceedings of the ACM SIGGRAPH Symposium*, Snowbird, UT, USA, pp167-77.

Soh, Chee-kiong and Soh, Ai-kah, 1990, A Knowledge-based Approach to the Design of Offshore Jacket Structure, *Computer-Aided Engineering Journal*, Feb., pp7-11.

Ssemakula, M.E., 1993, A Process Planning System for the CIM Environment, *Computer-intergrated Manufacturing Systems*, Vol.6, No.4, pp235-43.

Steinbock, Bill, 1991, 101 MDL Commands, OnWord Press.

St. Jacques, M., Stevens, D., Getchius, J. and Lau, L., 1992, A knowledge-based Method for Engineering Navigational Capability into User Interface

Software, *Proceedings Fourth International Conference on Software Engineering and Knowledge Engineering*, Capri, Italy, pp26-31.

Umaretiya, J. R. and Joshi, S.P., 1992, An Insight Into the Expert-Seisd: A Knowledge Based System for Structure Design, *Engineering with Computers*, Vol.8, No.3, pp151-161.

Unruh, V. and Anderson, D.C., 1992, Feature-based Modeling for Automatic Mesh Generation, *Engineering with Computer*, Vol.8, No.1, pp1-12.

Vander Zanden, B. and Myers, B.A., 1991, The Lapidary Graphical Interface Design Tool, *Human Factors in Computing Systems. Reaching Through Technology. CHI '91. Conference Proceedings*, New Orleans, LA, USA., pp465-6.

Wierda, Leo S., 1991, Linking Design, Process Planning and Cost Information by Feature-based Modelling, *Journal of Engineering Design*, Vol.2, No.1, pp3-15.

Xue, D. and Dong, Z., 1993, Fearure Modeling Incorporating Tolerance and Production Orocess for Concurrent Design, Concurrent Engineering: Research and Applications, Vol.1, No.2, pp107-16.

Zeid, Ibrahim, 1991, CAD/CAM Theory and Practices, McGraw-Hill.

Appendix A MDL Construction

MicroStation Development Language consists of (Intergraph, 1992, MDL Course Guide):

1. An implementation of the C programming language with some limitations and extensions (ANSI C).

2. A complete set of development tools: C compiler, linker, and librarian; Resource complier and librarian; make program; debugger.

3. A pseudocode interpreter inside MicroStation that executes MDL applications. The compiler compiles C source code into pseudocode that is understood by MicroStation.

4. A source-level debugger that is built into MicroStation.

5. A large runtime library with over 1200 functions. Most of these functions are "built-ins", which means that they are function within MicroStation itself. MDL simply holds pointers to these functions that applications may call directly. The advantage to "built-ins" is that they are in native machine code, and therefore run at full compiled speed. These functions are essentially the same functions that are used internally by MicroStation's commands. They are the heart of MDL.

6. Hooks that allow the application to modify MicroStation's behaviour. A hook is a pointer to a user-defined function. An application designates one of its functions as a hook and MicroStation calls that function when a certain event occurs.

7. Tools for developing a complete Motif-based Graphical User Interface (GUI). This interface can consist of dialog boxes, pull-down menus, and palettes.

8. A Resource Manager that controls data not actually included in the C source code. Examples of data include dialog box definitions and messages/prompts.

Appendix B Dialog Box Components

The standard dialog items and their main functions are listed as follows (Mach N Dinh-Vu,1991):

1.	Label:	draw a text string in dialog box.
2.	Group Box:	draw a rectangle with a 3D appearance around a group of dialog items.
3.	Toggle Button:	turn a variable's state on or off.
4.	Push Button:	activate a command with a push of button.
5.	Option Button:	select an option from a mutually exclusive set of choices.
6.	Scroll Bar:	a sliding bar that can change the value of a variable,between a maximum and minimum range.
7.	Text:	accept an input string.
8.	Colour Picker:	select a colour from a palette of 255 choices.
9.	Level Map:	select or change the on/off display state of MicroStation's design file level within a level map variable.
10.	Menu Bar:	used to create a strip of pull-down menus.
11.	Text Pull-Down:	a menu that contains a series of text strings.
12.	Option Pull-Down:	a pull-down menu with a list of options.
13.	Tool Palettes:	contain icons which, when activated, add a command to MicroStation's input queue.
14.	List Box:	allow the display of multiple text strings.
15.	Generic:	cerate items whose appearance and response to mouse and keyboard message can be customised.

Appendix C MDL File Types and Relationships MDL Utilities

MicroStation is delivered with support programs necessary to combine all required file into single application file. A application file has an extension of .MA and contains all the necessary data for the application to execute. The file types used to build applications are Include File (Header File), Resource, Source, Type and Make File (Intergraph, MDL Course Guide,1992).

Include File: These are standard C header files. They contain definitions shared by multiple source files.

Resource File : Resource files typically have and .RSC extension, and store data for use by an application. They are created by compiling resource compiler source files, which typically have an .R extension. Data stored in resource files including dialog box definition, user preference data, command table definitions, and messages to be displayed by an application. The Resource Manager controls access to the data structures stored in the resource file.

Source Files: Source files have an .MC extension. They store the C source code which is the heart of the MDL application.

Type Files: Type files have an .MT extension, and re used to generate type description of user-defined structures for evaluation by the C expression evacuator built-in functions in MDL.

Make Files: Make files have an .MKE extension. They are used to store the steps necessary to build files required to create an application. Makes files set up target files and dependent files. Target files can be created anytime any of the dependent files are dated later than the target.

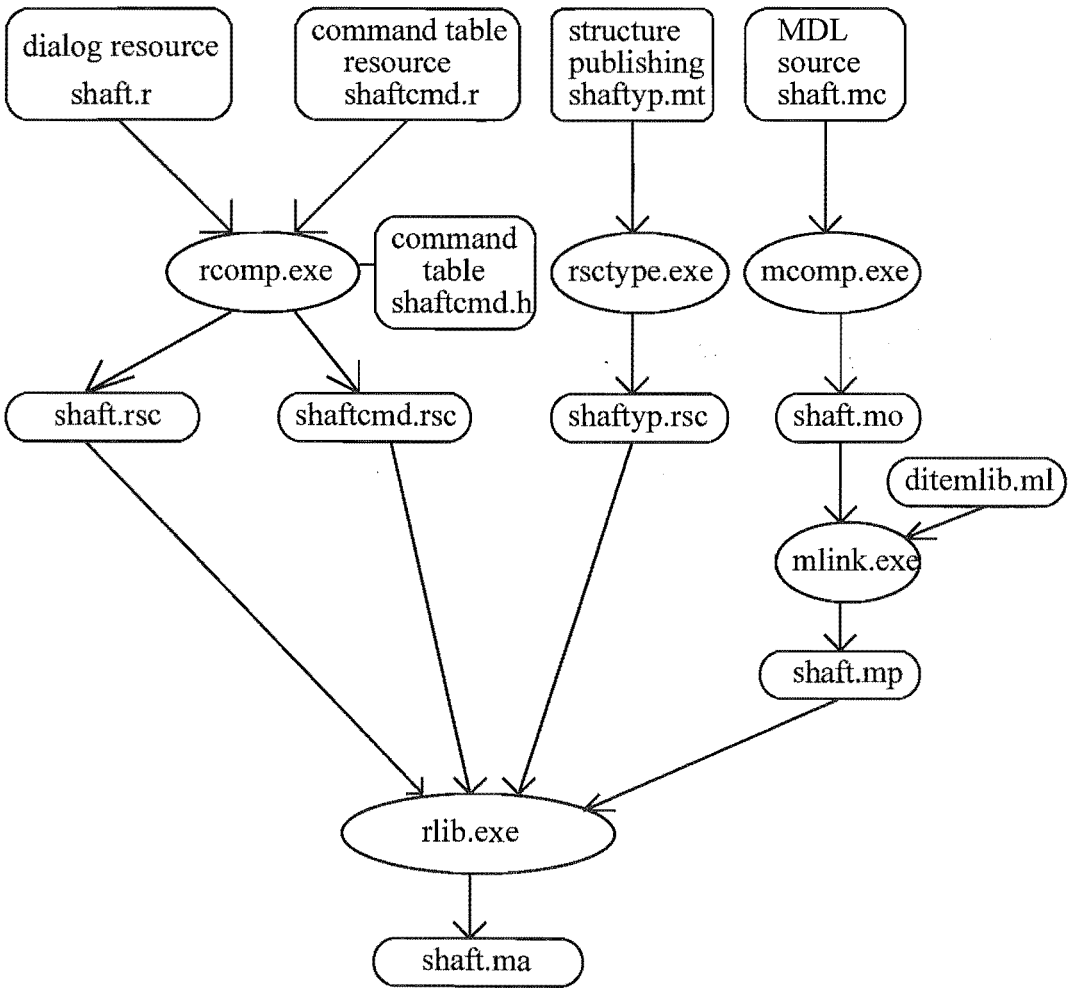
There are several files in the program. The files contain the following:

shaft.h	defines constants and data structures used by the application
shaft.r	contains the definitions of the application's dialog box resources

shaft.mc	contains the application's MDL source code
shaftcmd.r	contains the definitions of the application's command table resource
shaftyp.mt	used to generate type declaration resources for the application
shaft.mke	contains information used by BMAKE to create the application from the above files

File Relationship

The file relationships and program development cycle are shown in the figure blow.



MDL Utilities

They utilities programs used to create MDL applications are:

MCOMP - C source code compiler.
RCOMP - Resource compiler.
RSCTYPE - Type generator used to generate type definitions for a group of built-in functions that can be used to evaluate C expressions at runtime.
RLIB - Resource librarian used to merge multiple resource files into one resource file.
MLINK - Linker used to combine object files into a program.
BMAKE - "Make" utility used to automate compiling, linking, and resource building for application.

Appendix D Some Tips and Techniques in Dialog Box

1.1 Defining Icons

Icons can be included in Option Button or Option Pull-Down items as option choices. Icons are defined in Icon resource specification and also either in OptionButton resource specification or in Option Pull-Down resource specification. An example is shown as follows:

```
DItem_OptionButtonRsc OPTIONBUTTONID_Configuration =
        {

        SYNONYMID_Config,NOHELP,LHELPCMD,NOHOOK,ON|ALIGN_LEF
T,
        "",
        "info.config",
        {
        {Icon,ICONCMDID_Configuration,NOCMD,LCMD,0,NOMASK,ON,"0"},
        }
        };

  IconRsc  ICONCDDID_Configuration=
        {
         90,40,FORMAT_MONOBITMAP,BLACK_INDEX,
          "Configuration",
         {
         0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
         0x02,0x00,0x02,0x01,0x80,0x40,0x00,0x00,
          /* The rest of data that defines the icon */
           }
           };
```

This resource specification can be easily created and edited using an MDL application named " rasticon.ma" with the user definition option (Fig. D.1).

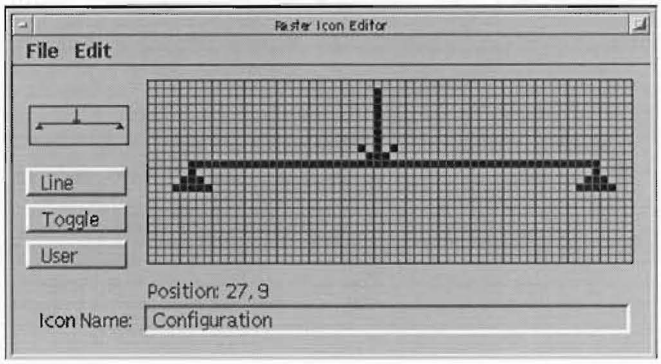


Fig.D.1 Icon Editor

1.2 Updating the Appearance of Dialog Items

The following code is listed to show how to update the appearance of a dialog box by using the mdlDialog_itemSynch function.

```
/* Open Shaft Information Dialog Box to put bearing reaction data*/
   mainbox_dbP= mdlDialog_open(NULL,DIALOGID_Information);
/* Find Text item for reaction data */
   dfP=mdlDialog_itemGetByTypeAndId(mainbox_dbP,RTYPE_Text,
         TEXTID_Reaction1H,0);
/* Update the Text item appearance */
   mdlDialog_itemSynch(mainbox_dbP,dfP->itemIndex);
    mdlDialog_itemSynch(mainbox_dbP,dfP->itemIndex+1);
    mdlDialog_itemSynch(mainbox_dbP,dfP->itemIndex+2);
    mdlDialog_itemSynch(mainbox_dbP,dfP->itemIndex+3);
    mdlDialog_itemSynch(mainbox_dbP,dfP->itemIndex+4);
    mdlDialog_itemSynch(mainbox_dbP,dfP->itemIndex+5);
    mdlDialog_itemSynch(mainbox_dbP,dfP->itemIndex+6);
    mdlDialog_itemSynch(mainbox_dbP,dfP->itemIndex+7);
```

1.3 Attaching A Hook Function to A Dialog Item

A sample of the process used to attach a hook function to a dialog item is illustrated as follows.

Firstly, hook functions are connected with a dialog box or item in Dialog or Dialog Item specification by the hook function ID which is defined in the header file. For example, for the user function - "draw_sketch" which is used to draw a shaft sketch with dimensions "# define HOOKITEM_Draw"

defines symbolic constants used as the hook function ID number. The hook function ID is attached to the "draw" PushButton in its item resource specification in the resource file (shaft.r). In the source file (shaft.mc), the DialogHookInfo array establishes the connection between hook function ID numbers and the user function address.

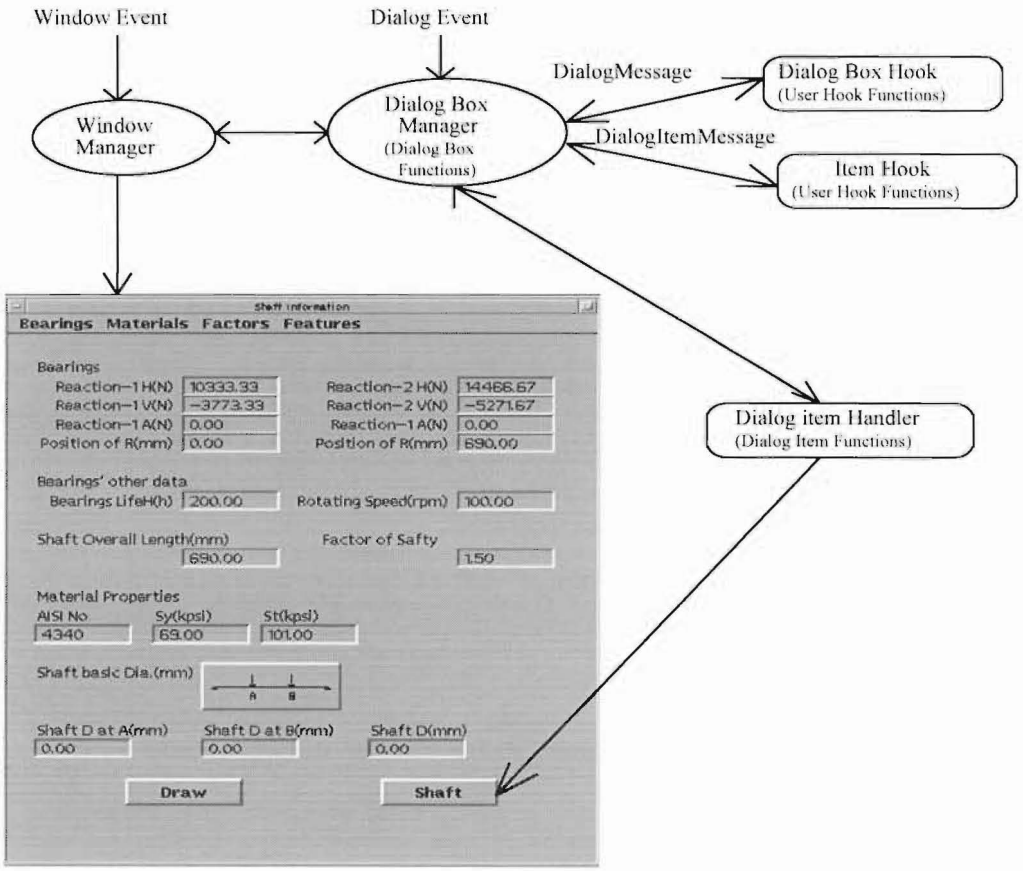
```
Private DialogHookInfo  uHooks[]=
{
  {HOOKITEMID_Draw, draw_sketch}
 }
```

Finally, the mdlDialog_hookPublish function call in the source file informs the dialog box manager of these connections. When the "draw" button is pushed , the program activates the function "draw_sketch".

Appendix E Dialog Hook Function

The dialog manger has three sub-systems that handle dialog boxes. These are **dialog box functions**, **item handler functions** and **user hooks**. The sub-systems communicate by sending **messages** of data structure to each other. The **dialog item handler** defines the default functionality and appearance of the dialog items (Intergraph, MDL manual, 1991, Mach N. Dinhg-Vu, 1991).

Since all dialog items have default behaviour, it is often desirable to modify the default action. Dialog hooks allow the programmer to attach user functions to either dialog boxes (**dialog hook function**) , or dialog items (**item hook function**).



The Dialog Manager collects information from the user and sends it to the MDL application via dialog hooks. The Dialog Manager reports on many events; it is the application's responsibility to sort out which events to process and which to ignore. A typical loop for a dialog box hook function is shown below.

```
Private Void dialogBoxHook
(
DialogMessage    *dmp
)
        {
         dimp->msgUnderstood =TRUE;
         switch (dmp->messageType)
           {
            case DIALOG_MESSAGE_CREATE:
                  : /* sent before item hooks are sent create messages */
                 break;
            case DIALOG_MESSAGE_INIT:
                  : /*sent after all item hooks are sent create messages*/
                 break;
            case DIALOG_MESSAGE_DESTROY:
                   :/* sent when the dialog box is about to be destroyed
*/
                 break;
            case DIALOG_MESSAGE_UPDATE:
                  : /* sent after the dialog manager updates dialog box.
*/
                 break;
            case DIALOG_MESSAGE_BUTTON:
                  : /* sent when a mouse button event occurs in a dialog
                    box */
                 break;
            case DIALOG_MESSAGE_KEYSTROKE:
                  :
                 break;
            case DIALOG_MESSAGE_CHILDDESTROYED:
                  :
```

```
            break;
        case DIALOG_MESSAGE_ANOTHEROPENED:
              :
            break;
        case DIALOG_MESSAGE_ANOTHERCLOSED:
              :
            break;
        default:
            dmp->msgUnderstood = FALSE;
          break;
      }
  }
```

Following is a typical loop for an item hook function.

```
Private Void itemHook
(
DialogItemMessage    *dmp
)
  {
  dimp->msgUnderstood =TRUE;
  switch (dmp->messageType)
   {
   case DITEM_MESSAGE_CREATE:
        : /* sent after the item is created */
        break;
   case DITEM_MESSAGE_INIT:
        : /* sent after item is created */
        break;
   case DITEM_MESSAGE_DESTROY:
        : /* sent when the item is about to be destroyed */
        break;
   case DITEM_MESSAGE_BUTTON:
        : /* sent when a mouse button event occurs in a mouse sensitive
           item */
        break;
   case DITEM_MESSAGE_QUEUECOMMAND:
```

```
        :
        break;
    default:
     dmp->msgUnderstood = FALSE;
        break;
    }
}
```

The first code sample reports on any action within the entire dialog box, while the second excerpt is only activated when a specific item is activated.

Appendix F Users Guide

This appendix is devoted to a user's guide for the shaft design assistant system. Guidelines for running the system are provided. The detail steps are demonstrated in combination with the examples in Chapter 8.

F1 Getting Started

(1) Prepare the shaft load and bearing data such as rotating speed, bearing basic rotating life in operating hours for the applications (Chapter 8.1).

(2) Enter MicroStation by opening a design file.

(3) Load Shaft Design Assistant System by keying in MDL L A:\shaft in the MicroStation Command Field in the Command Window. The Input Information Box then appears on the screen. Eg.

F2 Data Input

F2.1 Shaft Load Input (Fig.F2.1)

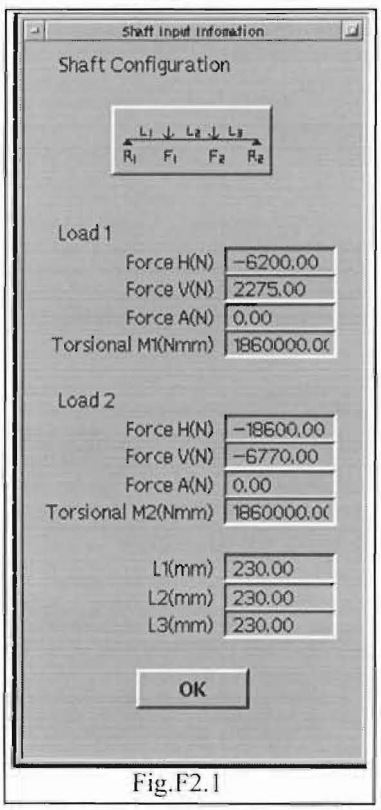


Fig.F2.1

(1) There is Option Button containing three options for shaft configuration. To select the layout which suits the application, move the mouse cursor over the icon and press and hold the Data button. Releasing the Data button selects the specific option for shaft configuration (Fig.F2.2).

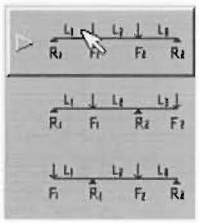


Fig F2.2

(2) The forces, torsional moments and length data for the shaft, which are shown on the shaft configuration icon, are then required. Using the mouse, Return and number keys data can be entered. Data can be edited after entry by using the same method.

(3) When data entry is completed, click the 'OK' button. The Shaft Information Box and Material Properties File Box will then open. The bearing reactions are calculated and displayed in the Shaft Information Box.

F2.2 Material, Bearing and Factor of Safety Data Input

(1) In the Material Properties Files Box (Fig.F2.3), select the material file by pointing at the material file name and pressing the Data button. Click the OK button. The materials List Box opens.

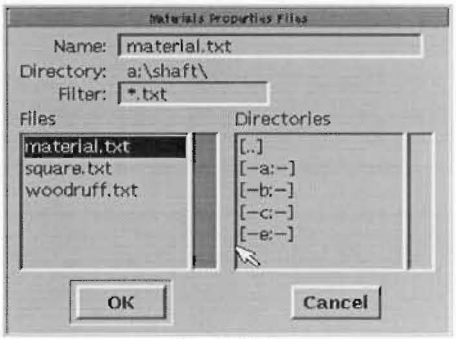


Fig. F2.3

(2) In the Material List Box (Fig.F2.4), select a material by pointing at one line and pressing the Data button. If necessary, scroll the list until a suitable material is visible by pointing in the scroll bar and pressing the Data button then click on Exit to close the box. Once selected, the material properties are displayed in the Shaft Information Box (Fig. F2.5).

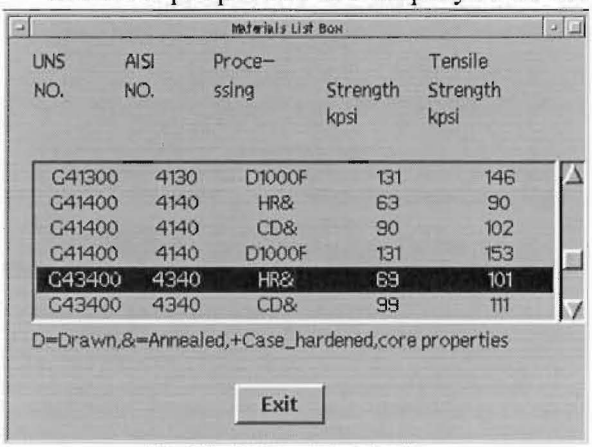


Fig.F2.4 Materials List Box

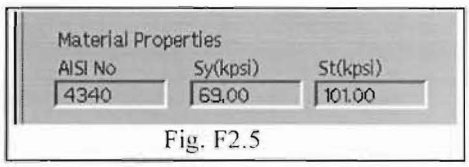


Fig. F2.5

(3) The bearing data can be entered in the text field named "Bearings' other data" in the Shaft Information Box using the method described in F2.1 (2) (Fig.F2.6).

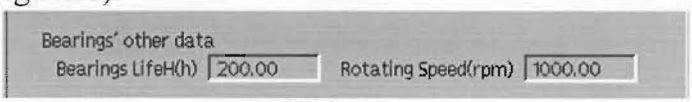


Fig.F2.6

(4) Select the factor of safety by pointing at Factors in the menu bar, pressing the Data button, and holding it down, the Factors pull down menu opens (Fig.F2.7). Drag the pointer to the factor required, then release the Data button to choose that factor. The factor value is displayed in the text field named 'Factor of Safety'. Alternatively, the data can be entered into text field directly (Fig.F2.8).

Fig.F2.7

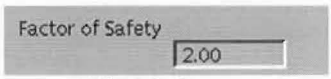


Fig.F2.8

F3 Shaft Layout

(1) In the Shaft Information Box, click the 'Draw' button (Fig.F3.1), the shaft layout with dimensions is then drawn in the MicroStation viewing windows (Fig.8.3).

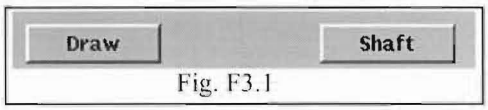


Fig. F3.1

F4 Shaft Diameter Calculation and Shaft Drawing

(1) In the Shaft Information Box, click the 'Shaft' button (Fig.F3.1), the shaft basic diameters are calculated based on failure under peak load and displayed in the text field named 'Shaft Basic Dia (mm)' (Fig. F4.1). The drawing with the basic diameter will then be shown in the MicroStation viewing windows (Fig8.4 (b)).

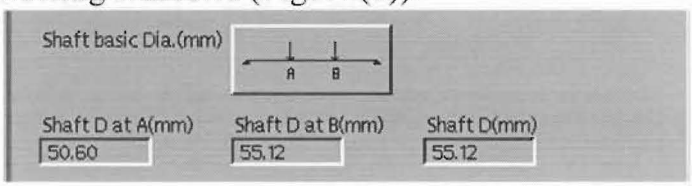


Fig. F4.1

F5 Shaft Detail Design

F5.1 Opening the Shaft Feature Selection Box

(1) Open the Shaft Feature Selection Box (Fig.F5.2) by pointing at "Features' in the menu bar on top of Shaft Information Box (Fig.F5.1), pressing the Data button, holding it down, dragging the pointer to 'Features' in pull-down menu and then releasing the Data button.

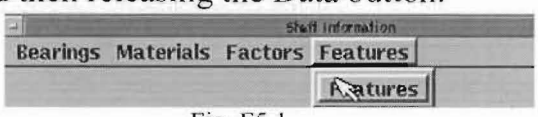


Fig. F5.1

The menu bar on the top of Shaft Feature Selection Box contains the names of shaft features such as Cylinder, Fillet, Bearing Seat, Key Seat and Spline which relate to the relevant feature description input box

(Fig.F5.2). The boxes currently in use are Cylinder, Fillet and Bearing Seat input box.

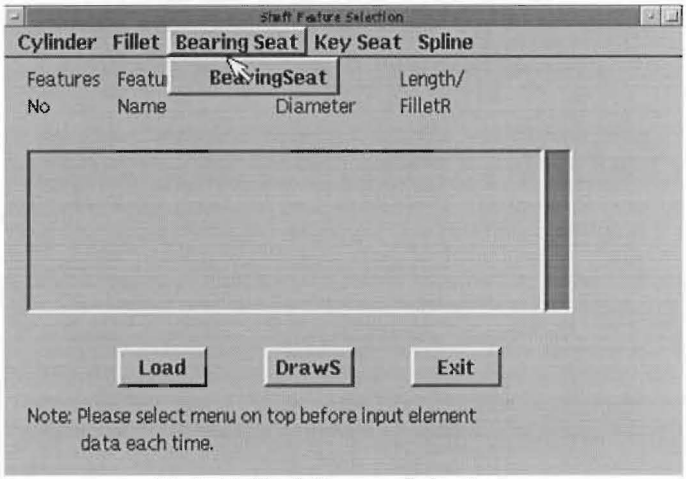


Fig.F5.2 Shaft Feature Selection

F5.2 General Steps for Shaft Feature Data Input

(1) Open one feature pull-down menu in the menu bar by pointing at a feature name in the menu bar, pressing the Data button and holding it down. Drag the pointer to the item in the pull-down menu. Then release the Data button to choose that item (Fig.F5.2). The relevant feature description input box will open.

(2) Feature parameters can then be entered.

(3) Click the "OK' when finished. The features data are then displayed in the string list field in the Shaft Feature Selection.

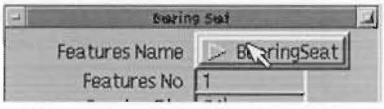
These steps are demonstrated as follows by taking the examples in Chapter 8.

F5.3 Bearing Seat Data Input

(1) Open Bearing Seat Input Box by using the method described in F5.2 (1). The Bearings Selection Box opens at same time (Fig.F5.3).

Fig.F5.3

(2) Drag the pointer to the Option button, press the Data button and release it to choose the feature name 'BearingSeat'. Key-in '1' in the text field named 'Features No' and press the Return key.



(3) Run the Bearings Selection System (a:\bearing\bearings.exe) by clicking the 'Run Expert System' button (Fig.F5.3). The Bearing Selection System main menu appears on the screen.

F5.3.1 Bearing Selection System

The bearing selection main menu is shown in Fig.F5.4. By using the Arrow keys to choose an item and pressing Return the selected program will be started.

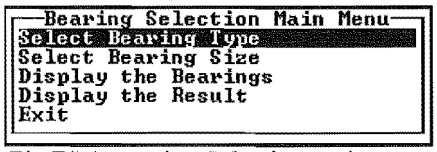
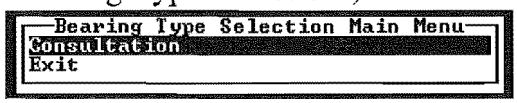


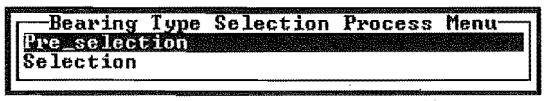
Fig.F5.4 Bearing Selection Main Menu

F5.3.1.1 Bearing Type Selection

(1) When choosing 'Bearing Type Selection', the following menu appears.



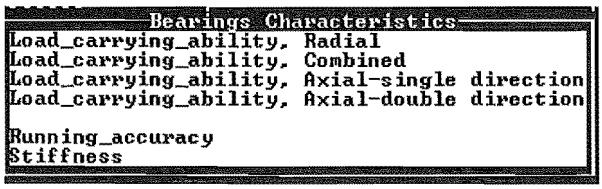
(2) Choose 'Consultant' to start the type selection process. A new menu appears as follows.



'Pre-selection' gives the bearing types available in the database.
'Selection' launches the type selection window.

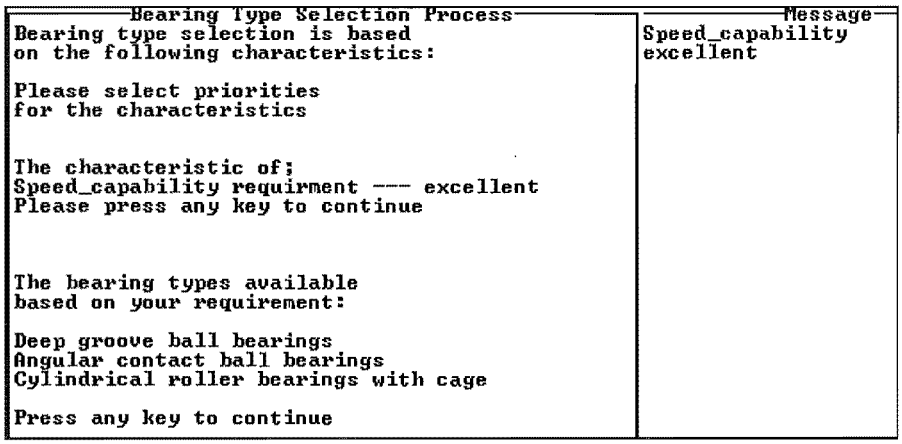
(3) When choosing 'Selection' in the above menu. The Bearing Type Selection Window appear as shown in Fig7.3.

(a) Use the Arrow and Return keys to choose the characteristic which is given an attribute priority according to its importance in the actual application. Eg. Speed-capability is given the top priority in the selection process.

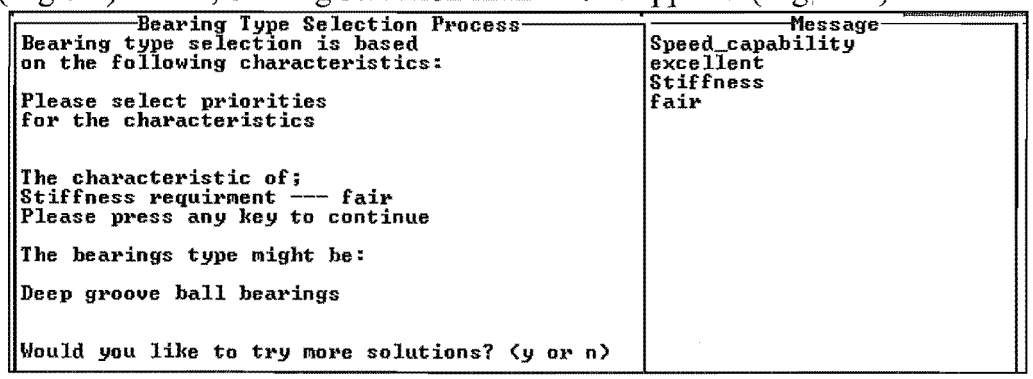


(b) Select a defining specification for the characteristic by using the Arrow and Return keys in the grade window. Eg. The requirement for speed - capability -- Excellent.

(c) The bearing types available based on the requirement appear on the screen.



(d) Repeat (a) and (b) to choose another characteristic until the one type of bearing is found. Eg. stiffness is given the second priority and graded as 'fair'. The bearing type which satisfies the requirement is Deep groove ball bearings. The program will ask 'If you would like to try more solutions?'. If yes, the Bearing Type Selection Process window appears (Fig.7.3). If no, bearing selection main menu appears (Fig.F5.4) .

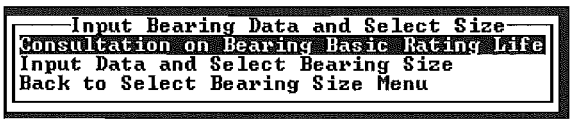


(c) If no result is found, the program asks the user to weaken or chose another specification and repeat the process ((a) - (d)).

F5.3.1.2 Bearing Size Selection

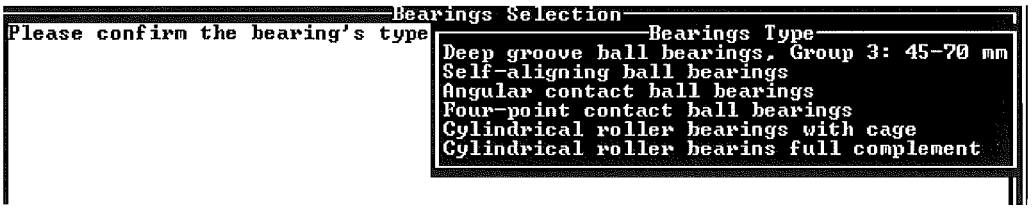
The following figure shows the menu for bearing size selection.

(1) When choosing 'Input Data and select Bearing Size', the new menu appears as shown below.

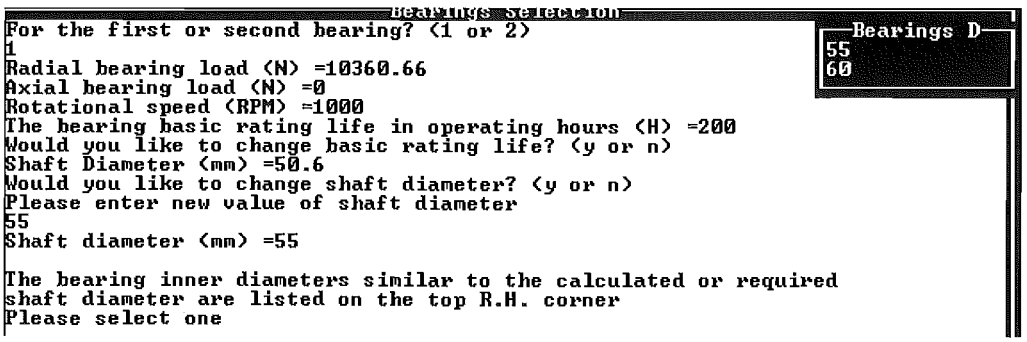


(2) 'Consultation on Bearing Basic Rating Life' provides consultancy on bearing basic rating life in operating hours in different usages. The Bearing Selection window appears after consultancy.

(3) 'Input Data and Select Bearing Size' opens the Bearing Selection Window with a list of all bearing types on the screen. Use the Arrow and Return keys to confirm one. This will help to narrow the database which should be searched after the bearing data is entered. A question appears on the screen to check if the selection is for the first or second bearing. The bearing data is read from the disc file (a:\bearing\shaftb1.dat or shaftb2.dat) which is output from MicroStation and displayed in the window. The shaft diameter and bearing basic rating life can be changed at this stage (Fig.7.4 and Fig.8.6).



(4) The bearing inner diameters similar to the calculated or required shaft diameter are listed on the top right hand corner. Using the Arrow and Return keys select one.



(2) Shaft data can be retrieved from the disc file (A:\shaft.rsc) by clicking the 'Load' button.

F8 Unload Shaft Design Assistant System

(1) Unload Shaft Design Assistant System by keying-in 'MDL unload shaft' in the command field of the Command Window.

F9 DXF File to Algor and MasterCAM (Fig.8.10, Chapter 6.1)

This part is discussed in Chapter 6.1.

(5) The available bearings which satisfy the requirement are shown in Fig.8.7 (a). The basic rating life of the bearings are listed in the window. One bearing can be selected by the user.

F5.3.1.3 Display the Bearings

This part aims at providing the bearing groups available in the database. This section requires more programming so the database contains a full range of bearing data for different bearing types.

F5.3.1.4 Display the Result

This part shows the finally selected bearing. Eg. Fig.8.7 (a).

F5.3.1.5 Exit the System

Exiting the Bearing Selection System is achieved by choosing 'Exit', the question 'Are you sure?' appears on the window. If 'yes', the system will write the result to a disc file (a:\bearing\bearing.dat) for later use in MicroStation.

F5.3.2 Read the Bearings Data

(4) When exiting the Bearings Selection System, the screen returns to MicroStation automatically.

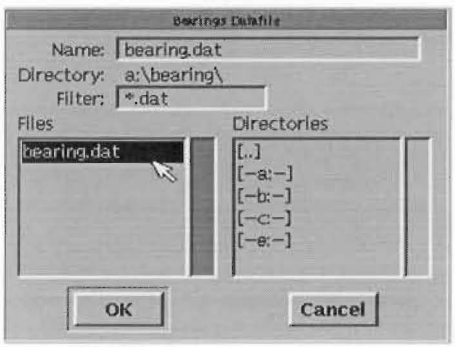
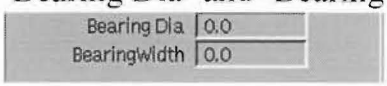


Fig.F5.5

(5) Click 'the Read Data File' button. The Bearing Data File box (a:\bearing*.dat) opens (Fig.5.5). In the box, select the bearing data file by pointing at the file name and pressing the Data button. Click the OK button. The bearing data is then displayed in the text field named 'Bearing Dia' and 'Bearing Width'.



(6) Click the "OK' button. The bearings seat data appears in the Shaft Feature Selection Box (Fig.5.6).

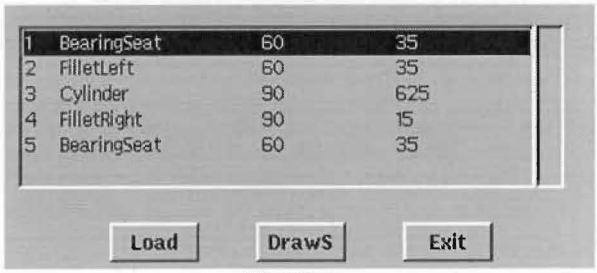


Fig.F5.6

F5.4 Fillet and Other Features Data (Fig.8.9 (a), (b))

(1) Open the Fillet Input Box by using the method described in F5.2(1).

(2) Drag the pointer to the Option button, press the Data button and release it to choose the feature named 'FilletLeft'.

(3) Key-in '2' in the text field named 'Features No'. Enter data in the text field named 'Diameter' and 'Fillet Radius'. Press the Return key when each text field is completed.

(4) Click the 'OK' button. The fillet data appears in the Shaft Feature Selection Box.

(5) Enter Cylinder and FilletRight data by using the method described in F5.4.

(6) Enter the second bearing seat data by using method described in F5.3 or F5.4.

(7) Fig.F5.6 shows the whole shaft data.

F6 Shaft Drawing

(1) The shaft draw action is effected by clicking the 'DrawS' button (Fig.F5.6).

(2) The message - 'Enter shaft origin' appears in the Message Field in the MicroStation Command Window. Click one point in one of the viewing windows to define the shaft origin. The shaft drawing is then completed in the viewing windows.

F7 Save and Load Shaft Data

(1) Shaft data is saved on the disc file (A:\shaft.rsc). The function is automatically effected when the 'DrawS' button is clicked.

Appendix G Shaft Design Knowledge

G1. Shaft Design for Static Loads

The design method used to calculate a preliminary shaft diameter in the Shaft Diameter Calculation and Drawing module is based on static loads. The formula comes from Shigley's Mechanical Engineering Design (Shigley, 1983, p689).

The stress at the surface of a solid round shaft subjected to combined loading of bending and torsion are:

$$\sigma_x = \frac{32M}{\pi d^3} \qquad \tau_{xy} = \frac{16T}{\pi d^3}$$

Where σ_x = bending stress
τ_{xy} = torsional stress
d = shaft diameter
M = bending moment at critical section
T = torsional moment at critical section

By the use of a Mohr's circle it is found that the maximum shear stress is

$$\tau_{max} = \sqrt{\left(\frac{\sigma_x}{2}\right)^2 + \tau_{xy}^2} \qquad \text{(a)}$$

Eliminating σ_x and τ_{xy} from Eq. (a) gives

$$\tau_{max} = \frac{16}{\pi d^3}\sqrt{M^2 + T^2} \qquad \text{(b)}$$

The maximum-shear-stress theory of static failure states that $S_{sy} = S_y / 2$. By employing a factor safety n, we can now write Eq. (b) as

$$\frac{S_y}{2n} = \frac{16}{\pi d^3}\sqrt{M^2 + T^2} \quad \text{or} \quad d = \left[\left(\frac{32n}{\pi S_y}\right)\left(M^2 + T^2\right)^{1/2}\right]^{1/3}$$

Where S_y = yield strength in tensile
S_{sy} = yield strength in shear

G2. Selection Bearings (SKF General Catalogue, 1970 and A G Herraty, New Methods for the Selection of Rolling Bearings, Selecting Bearings for Economical and Reliable Designs, Mechanical Engineering Publications Limited for Publication Services, 1987)

G2.1 Selection of Bearing Type

The selection of bearing type is made by reference to the Table G1 in which the characteristics of each bearing type are evaluated in general terms.

G2.2 Selection of Bearing Size

The equivalent dynamic load, P, acting on the bearing is calculated from the radial and axial bearing loads (Fr and Fa) using the formula:

$$P = xFa + yFy$$

where x,y : factors and can be found in manufacturer' literature
P: the equivalent dynamic load
Fa: axial bearing loads
Fr: radial bearing loads

The relationship between the basic rating life, the basic dynamic load rating and bearing load is expressed by the equation:

$$L_{10h} = \frac{1\ 000\ 000}{60\ N}\left(\frac{C}{P}\right)^{p}$$

where L 10h : basic rating life in operating hours
n: rotational speed, r/min
C: basic dynamic load rating, N
P: equivalent dynamic bearing load, N
p: exponent for the life equation,
p = 3 for ball bearings
p = 10/3 for roller bearings

The selection of bearing size is made by comparing the life calculated with the life required for the application. The typical life time in operating hours for common engineering equipment are included in the program and can be used for guidance. If the life calculated is more than the required life the bearing is satisfactory.

The bearing database in the program contains only limited bearing data for deep groove ball bearings. The factors, x,y, for equivalent bearing load are taken from the Table G2 below.

Table G2 Calculation Factors for Deep Groove Ball Bearings

		$F_a/F_r \leq e$		$F_a/F_r > e$	
F_a/C_0	e	x	y	x	y
0.025	0.22	1	0	0.56	2
0.04	0.24	1	0	0.56	1.8
0.07	0.27	1	0	0.56	1.6
0.13	0.31	1	0	0.56	1.4
0.25	0.37	1	0	0.56	1.2
0.5	0.44	1	0	0.56	1

C_0 = static load rating e= specified value

Table G1 General Characteristics of Bearing Types

Bearing Type \ Characteristics	Load Carrying Ability			Speed capability	Running accuracy	Stiffness	Quiet operation	Low torque	Suitable as locating bearing	Suitable as non-locating bearing
	Radial	Axial	Combined							
Deep groove ball bearings	+	<+>	+	+++	+++	+	+++	+++	++	+
Self-aligning ball bearings	+	<->	-	++	++	+			+	+
Angular contact ball bearings	++	<++	++	+++	+++	++	++		++	
Four-point contact ball bearings	+	<++>	++	+	+	++			++	
Cylindrical roller bearings with cage	++	--	--	+++	+++	++	+	++	--	++
Cylindrical roller bearings full complement	+++	<-	-	-	++	+++			-	
Needle roller bearings	++	--	--	++	++	++			--	++
Alignment needle roller bearings	++	--	--	++	+	+			--	++
Spherical roller bearings	+++	<+>	++	++	++	++			++	+
Taper roller bearings	+++	<++	++	+	++	++			+++	
Thrust ball bearings, single direction	--	<+	--	+	+++	++				
	--	<+	--	+	++	++				
Thrust ball bearings, double direction	--	<+>	--	+	++	++				
Cylindrical roller thrust bearings	--	<++	--	-	+++	+++				
Needle roller thrust bearings	--	<++	--	-	+	+++				
Spherical roller thrust bearings	-	<+++	+	-	++	++				

Symbols:

+++	excellent	<	single direction
++	good	< >	double direction
+	fair		
-	poor		
--	unsuitable		

APPENDIX H

Map of Communication in the Local Environment